Parameter Reduction in Deep Learning and Classification

David Browne
BPHYS
110715117

Thesis submitted for the degree of Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
INSIGHT CENTRE FOR DATA ANALYTICS

February 2020

Head of Department: Prof Cormac J. Sreenan

Supervisor: Dr Steven Prestwich

Contents

List of Figures

List of Tables

I, David Browne, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

David Browne

Abstract

"If I had eight hours to chop down a tree, I'd spend the first six hours sharpening my axe."

Abraham Lincoln

The goal of this thesis is to develop methods to reduce model and problem complexity in the area of classification tasks. Whether it is a traditional or a deep learning classification task, decreasing complexity helps to greatly improve efficiency, and also adds regularization to the models. In traditional machine learning, high-dimensionality can cause models to over-fit the training data, and hence not generalize well, while in deep learning, neural networks have shown to achieve state-of-the-art results, especially in the area of image recognition, in their current state cannot be easily deployed on memory restricted Internet-of-Things devices.

Although much work has been carried out on dimensionality reduction, the first part of our work focuses on using dominancy between features in the aim to select a relevant subset of informative features. We propose 3 variations, with different benefits, including fast filter features selection and a hybrid filter-wrapper approach. In the second section, dedicated to deep learning, our work focuses on pruning methods to extract an overall much more efficient neural network.

We show that our proposed techniques outperform previous state-of-the-art methods, across the different classification areas on a number of benchmark datasets using various classifiers and neural networks.

To beloved ones...

Acknowledgements

From the start of my PhD study at the Insight Centre for Data Analytics, at University College Cork, there have been many people, both staff and fellow students, who have helped me in different ways. I cannot thank them one-by-one, but for any not personal mentioned below, I offer my sincere gratitude for their help.

First of all, I would like to express my utmost appreciation to my supervisor Dr. Steven Prestwich, for his guidance, encouragement and thoughtful insights throughout my research. I am very grateful for the opportunity he gave me for both studying under him and working with him. Through Steve's guidance with proper feedback, both praise and constructed criticism, I have learnt so much, and has led me to a better direction in my research.

I would specially like to thank Steve in taking a chance of allowing me to research Deep Learning, with the industry support of United Technology Research Centre. This leads me to offering my thanks to Michael Geiring, a strong influence on my deep learning study direction. Both Michael and Blanca, both from UTRC, help me start my study into deep learning, and guided me over the initial bumps in the road, paving a path for my keen interest in the area. Their and Steve's support, chat and intense discussions helped me a lot throughout my research in the area, and still fuels my desire for deep learning.

I would like to offer my gratitude to Sebastian Scheurer, who shared his knowledge on WiFi analytics, that helped me understand the data better in order to be able to carry out my research in that particular area. I would also like to thank a past post-doc Carlo Manna, who partnered with me, helping to guide me on my first publication, and gave me good advice that is still relevant for me today.

I would also like to thank to other members of the Insight Centre for Data Analytics. Especially, Federico Toffano, who worked with me on different ideas for so many weekends. For interesting chats, both research related and off-topic, my thanks to Ali Naeem and Daniel Desmond. My gratitude also goes to Andrea Visentin, Sorina Chisca and Diego Carraro.

I would like to offer a sincere appreciation to my partner Rose O'Dwyer, sister Jacinta Browne and her husband Andrew Fagan, who are the main reasons I am here today. It was their words of wisdom and encouragement, on a late night in Dublin city in 2009, that put me on this path to research. Infact, it was Jacinta who practically organised my interview for St.Johns College Cork to get my foot on the ladder to my studies. And from there, it was through Rose's support that helped me carry on my under-graduate Physics studies, especially during the hard times. All 3 of them have been a rock to me in all aspects of my studies, and continue to be throughout my PhD research.

Another very special mention is to my little princess Arianna Reilly who, un-

knowingly, motivates and drives me to achieve the best I can. More importantly, when my research gets tough and I feel like I cannot do it or there is no end to it, she always manages to make me smile and laugh, re-energizing my motivation.

Finally, I am very grateful that my family encouraged me throughout my studies. My mother and father deserve special gratefulness for their support, and their continuous "have you wrote much of your thesis yet?" inputs!!

This quote sums why I returned to study:

"*Twenty years from now you will be more disappointed by the things that you didn't do than by the ones you did do.*"

Mark Twain

Chapter 1

Introduction

This thesis focuses on reducing the complexity of classification problems, in order to improve the computational efficiency and the accuracy of the classifiers. By decreasing the number of parameters, and features, within the data and models, not only do we create computationally and energy efficient models but also help to counteract over-fitting. One of the most important steps in the pre-processing stage of a machine learning project is feature selection. In chapters $2 - 4$ we look at feature selection, which selects the most informative features and removes the irrelevant and redundant ones. In this part of the work, we look at credit scoring and microarray data, and use standard machine learning classifiers, including but not limited to *Logistic Regression, Naive Bayes, Random Forest, Support Vector Machines*. Feature selection not only improves model efficiency, but in this work we show it was used to predict the most important influencing variables in both credit scoring data (continuous, ordinal and nominal) and microarray data (genes).

The next three chapters $5 - 7$ focus on image recognition classification using Deep Learning Convolutional Neural Networks. The algorithms proposed in the feature selection part of this work were unsuitable for deep learning due to being computationally too expensive because they used the data labels to determine the important features. This type of supervised approach would significantly increase the training time of the deep learning networks. Therefore, instead of removing features of the raw data, we look within the neural networks, both at the feature maps and filters/nodes to remove (or prune) redundant parts of the network. We run experiments on image data using well-known networks; *AlexNet, VGG16, CifarNet*. Next we give a brief introduction to each

chapter.

In chapter 2, we introduce a filter feature selection method named *Relevance-Redundancy Dominance* (RRD), which is based on the idea of dominant features being the most informative. Thus, by keeping the dominant (relevant) features, and using them to remove the dominated features (redundant), along with ranking them in order of importance with respect to the target, we extract an optimal minimum subset of features. This novel filter features selection algorithm is performed in an unsupervised manner, meaning that, unlike similar approaches in this research area, there is no threshold limit for the dominance or correlation, or pre-selected number of features to retain.

We carry out extensive experiments on benchmark credit datasets, and compare and contrast the results with current state-of-the-art (SOTA) results in the field. We choose credit scoring datasets due to having a mixture of data types e.g. continuous, ordinal and nominal, and also due to the possible impact in the financial area. Bad loan decisions cost the credit industry vast amounts each year, therefore, machine learning algorithms are used to develop credit scoring models that help the industry to classify a new client as a good risk or a bad risk using the client's information such as age, sex and purpose of loan [CH03]. Generally, finance and banking institutes screen customers before authorising a loan using predictive models, combined with domain knowledge. These models are usually trained on historical customer data in a supervised manner, and a very important part of the model training is selecting important variables while neglecting the *noisy* variables. If the variables selected are not relevant or redundant features are selected, the model's classification power will be significantly reduced, which could cost the institutes customer loyalty for false negatives or monetary value for the false positives.

Further tests are performed using *RRD* on higher dimensional datasets to check the robustness of the method. Initial tests on both pharmaceutical and microarray data, showed promising results, but experiments using a larger range of microarray data showed, that although classification accuracy was within an acceptable threshold, *RRD* had a tendency to retain a greater number of features than required which, *(i)* increased the time needed for the algorithm to select its subset of features, and *(ii)* can cause the classifier to over-fit and not be as efficient as possible. We offer a solution for the short-comings of *RRD* in the next chapter, with an adapted variation called *Fast Relevance-Redundancy Dominance* (FRRD), which aimed to rapidly find the optimal minimum subset

of features on high dimensional data.

We propose an extension of the work of *RRD*, in chapter 3, with an algorithm we call *Fast Relevance-Redundancy Dominance* (FRRD). The main difference between algorithms are; *FRRD* first, in a very fast manner, reduces the number of features in the dataset, subsetting it, only keeping *n* of the most dominant features. The user pre-selects the value of *n*, which in all the experiments in this work was set to the same value. Iteratively, *FRRD* selects the top most correlated remaining feature with respect to the target, F, and removes the features dominated by F. This is repeated until we have the desired number of remaining dominant features. Step 2 of the algorithm is performing *RRD* on these remaining features, and the output of this is the relevant, non-redundant, un-dominated subset of features for the classifier.

To analyse the robustness, scalability and competitiveness of our proposed algorithm, we carry out various types of experiments on high dimensional data (microarray data), that ranges in the number of features from 2000 to 24188. We compare *FRRD* classification accuracy, along with its stability and scalability, against SOTA feature selection techniques using the same experimental setup on a range of microarray datasets. Microarray datasets are commonly used in cancer clinical studies, where the identification of the most informative genes to detect cancer is an important biological challenge. Due to the high dimensionality of microarray data, the ability to find a subset of features with high potential bio-markers and strong candidates that are connected with the disease among the vast number of genes in the samples, is a vital part of research in this area. Therefore, an algorithm that can find these relevant genes, and help build an accurate classifier to distinguish between cancerous and noncancerous samples, is an important machine learning feature selection task.

Although *FRRD* is a highly scalable feature selection method, which helps to develop accurate classifiers, its main shortcoming is that the user has to predefine the number of dominant features that step 1 of the algorithm retains. To improve upon both *RRD* and *FRRD*, we put forward an improved version, still based on the idea of dominance, but in conjunction with a genetic algorithm.

In our final feature selection chapter 4, still in the area of data dimensionality reduction, we introduce *improved Relevance-Redundancy Dominance with a Genetic Algorithm* (iRRD-GA). Again, we analyse microarray data, as we show our improved algorithm takes the benefits of both a filter and wrapper feature selector. The new filter feature selection part of iRRD-GA uses an improved

unsupervised version of the algorithm proposed in chapter 2, which uses 2 very different statistics to select minimum subsets, giving the dominant genetic algorithm more diverse selection to retain a *good* final subset for the classification part. The genetic algorithm (GA) part of iRRD-GA is a tournament style GA, where the 2 dominant genes selected in the process are considered to be the parent genes.

This is where the traditional feature selection to reduce problem complexity ends, and we propose methods to improve deep learning networks by reducing their complexity in the following 3 chapters. The previous proposed approaches, although they had great success in the area of dimensionality reduction, required the use of the target values to evaluate the selection of subsets which would be too computationally expensive in the area of deep learning.

Chapter 5 is the beginning of the Deep Learning section of the Thesis, and it is where we introduce Pulse-Net, our novel pruning method for convolutional neural networks (CNN). A CNN is one of the most prominent deep learning approaches, and it has pushed the boundaries in the state-of-the-art image recognition results. Although CNNs were being used on various tasks by the turn of the 20th century, it wasn't until 2012, when AlexNet shattered current image classification methods on the well-known ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for that year, did they increase significantly in popularity. The advancements in Graphic Processing Units, has helped push the surge of interest in this research area, with their ability to perform matrix multiplication very efficiently, which is due to both memory bandwidth and parallelism. In many ways these CNNs are similar to the older neural networks, multi-layer perceptron, as they both are trained by tuning weights and biases, but differ as CNNs use convolution calculations to consider patches of the input (image) to be spatially related.

Considering CNNs, main usage is on image classification problems, this is the area we choose to implement our proposed algorithm Pulse-Net, and ran experiments using 3 benchmark datasets in this area. We analyse our algorithms' robustness on 3 popular CNNs and compare the results to current state-of-the-art pruning and compression methods. It has been well documented that neural networks are over-parameterized, which is especially true once the network has been trained. Since the fundamental building blocks of a neural network, artificial neurons, are based on the brain's neurons, we will use this as a way to picture the reasoning. In a child's brain, there are millions of these neurons

interconnected to soak up all the information and allow the child to learn, but as they get older many of these neurons become dormant or switched off as less learning is required. Following the same principle, we allow a CNN to be over-parameterized and allow it to fully converge through training. Once no more training improves its classification accuracy, like in the brain, we prune what we see as the irrelevant parts and allow the network to fine-tune to recover afterwards.

This is the basis of the idea behind our proposed algorithm, which has application purposes because of it being highly efficient, allowing it to run on memory restricted Internet of Things devices. We demonstrate this theoretically by showing how Pulse-Net improves the computational inference speed, while reducing its storage overhead and energy usage on various commonly used CNNs. Although we show theorectically that the produced pruned networks are suitable for Internet of Things devices, showing this practically warrants further investigation. As an additional interesting set of tests, we explored if this idea of pruning a network helps to create a network that is more robust against adversarial attacks. An adversarial attack is where a test image is subtly modified in such a way that it is almost undetectable to the human eye. The hypothesis was that, the less parameters the network had, would make it less prone to an adversarial attack, as these types of attacks focus on the weights of the network.

The next chapter, 6, is a continuation of the Pulse-Net research. We use the same datasets and CNNs to compare and contrast both pruning methods. One of the main differences, amongst others, is that our new improved version named *unsupervised PulseNet* uses an intelligent pruning decision approach. The clustering algorithm k-means has been shown to be a powerful unsupervised method to group similar samples into clusters, and it is this property that we exploit in unsupervised PulseNet. We deem only one filter/node in the cluster to be relevant and the rest of the filters/nodes are pruned, considered to be redundant. There are other significant changes in unsupervised PulseNet compared to the original version, but we believe that this selection of a more intelligent and natural way of pruning the network helps us achieve a better compression-to-accuracy ratio. Also, unlike the original PulseNet, this improved version benefits from faster training due to the extraction of the smaller refined network at each pruning iteration.

Although unsupervised PulseNet showed great success to reduce model complexity, improving the network efficiency, its main pitfall is that to achieve this

better compression-to-accuracy ratio requires a number of pruning and fine-tuning iterations. Our final variation of PulseNet is called PulseNetOne, a pruning method that because it only requires a single shot at removing redundant filters/nodes is a much faster version. We consider this version to be more industry applicable, while the other versions were more theoretically appropriate showing how far we can compress a network with minimum accuracy loss. Since PulseNetOne is focused on being more application based, we selected a research area where evolving efficient CNNs is yet to be explored. We choose six benchmark remote-sensing datasets to display the effectiveness of PulseNetOne on both AlexNet and VGG16, and compare to the current state-of-the-art. Our main expected result was to significantly reduce the network's complexity, extracting a smaller and overall more efficient model, but this reduction of network parameters also improved the model's classification accuracy due to regularization of the network. In chapter 7, we also take advantage of transfer learning, similar to much or the related work in this area, due to the datasets having a limited amount of training data.

1.1 Contributions

This thesis has 2 main contributions, both of which have a number of versions which, depending on the task can be implemented. The first is the filter feature selection algorithms based on Relevance, Redundancy and Dominance. These set of algorithms demonstrated that by reducing the number of features in a dataset, helped to reduce over-fitting, making the classifiers more robust. The algorithms were evaluated on both credit scoring data and microarray data. In the area of credit scoring, the proposed algorithms can reduce monetary loss, while in the area of microarray data, any improvements in assisting doctors to help detect cancerous samples is invaluable.

The second contribution consists of different methods to reduce the size of deep learning networks, through pruning. We proposed an iterative unsupervised approach, which achieved SOTA compression with minimum accuracy loss. We also introduced a fast version of this pruning approach, which could be more suitable when there is limited time to find a fully compressed network. These pruned networks are ideal for memory restricted devices, which are currently very popular in the area of smart buildings and Internet-of-Things.

Due to deep learning being a very hot topic currently, we believe our PulseNet

pruning algorithm to be our most important, and also our favourite, contribution.

1.2 Publications and Submissions

We have published and submitted the work described in this thesis in a journal, international conferences, their workshops and poster sessions. These publications are listed as follows:

- David Browne, Carlo Manna and Steven Prestwich: *Relevance-Redundancy Dominance: a threshold-free approach to filter-based feature selection*. 24th Irish Conference on Artificial Intelligence and Cognitive Science 2016.
- David Browne, Carlo Manna and Steven Prestwich: *Fast Relevance-Redundancy Dominance: Feature Selection for High Dimensional Data*. Multi Conference on Computer Science and Information Systems, pages 255-262, 2017.
- David Browne, Michael Giering and Steven Prestwich: *Pulse-Net: Dynamic Compression of Convolutional Neural Networks*, IEEE 5th World Forum on Internet of Things (WF-IoT), pages 346-351, 2019.
- David Browne, Michael Giering and Steven Prestwich: *Unsupervised PulseNet: Automated Pruning of Convolutional Neural Networks by K-Means Clustering*. Submitted to 2020 European Conference on Computer Vision (ECCV 2020).
- David Browne, Michael Giering and Steven Prestwich: *PulseNetOne: Fast Unsupervised Compression of Convolutional Neural Networks for Remote Sensing*. MDPI Journal Special Issue "Lightweight Deep Neural Networks for Remote Sensing Image Understanding".
- David Browne, Carlo Manna and Steven Prestwich: *Gene selection for microarray data using hybrid dominance optimization*. Submitted to The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2020).

1.3 Thesis Structure

The structure of the thesis is:

- In Chapter 1, we have presented the motivation behind our work and we have summarized the contributions that we have made.
- In Chapter 2, we introduce Relevance-Redundancy Dominance, showing its effectiveness on credit scoring problems.
- In Chapter 3, we introduce Fast Relevance-Redundancy Dominance, showing its effectiveness on high dimensional microarray problems.
- In Chapter 4, we introduce improved Relevance-Redundancy Dominance with a Genetic Algorithm, showing its effectiveness on high dimensional microarray problems.
- In Chapter 5, we present Pulse-Net, a pruning method showing its robustness on image recognition tasks.
- In Chapter 6, we present unsupervised PulseNet, an intelligent pruning approach and show its robustness on image recognition tasks.
- In Chapter 7, we present PulseNetOne, a one shot pruning approach and report its compression-to-accuracy ratio on remote-sensing and scene image recognition tasks.
- In Chapter 8, we conclude the thesis and offer ideas for future work.

Chapter 2

Relevance-Redundancy Dominance: Threshold-Free Filter-Based Feature Selection

2.1 Motivation

In this chapter, we introduce our novel algorithm, based on dominance, to help us select the relevant features in credit datasets, or using machine learning terminology feature selection. Feature selection is used to select a subset of relevant features in machine learning, hence reducing the dimensionality which is vital for simplification, improving efficiency and reducing over-fitting. The dimensionality of the dataset is the total number of independent variables, or features, the data has. A threshold is typically used to limit the set of selected features, and features can also be removed based on similarity to other features (redundancy). Some methods are designed for use with a specific statistic. We present a new filter-based method that can be applied to numerical, categorical, binary and a combination of all 3 data types, can use a variety of statistics, evaluates features in terms of relevance and redundancy, and does not require a threshold. It does not have to calculate the full correlation matrix between feature-to-feature, hence, it is more computationally efficient than other similar approaches, mRMR [DP05] and CFS [Hal99a], and as it considers the relationship between features it is less likely to have correlated features in the subset than FCBF [YL03a] would have. The robustness of RRD is shown as it outperformed published algorithms on credit scoring, and from initial results shown, it

can also be used very effectively on high-dimensional data, such as microarray data.

2.2 Introduction

Many real-world applications deal with high-dimensional data. Feature selection has a key role to play in helping reduce high-dimensionality in machine learning problems. Removing unimportant features reduces data size, and improves learning accuracy and comprehensibility. Nowadays 5 exabytes of data is produced every 2 days, and the pace of production continues to rise [BCAB18]. Because the volume, velocity, variety and complexity of data sets is continuously increasing, feature selection techniques have become indispensable in order to extract useful information from huge data sets.

By means of feature selection techniques, attributes that allow a problem to be clearly defined are selected, while irrelevant or redundant data are ignored. Feature selection methods have traditionally been categorized as *filter*, *wrapper* or *embedded* methods [GGNZ08], though new approaches that combine existing methods or are based on other machine learning techniques are continuously being developed, including recent developments in methods for high-dimensionality problems in areas such as clustering [CH03, SNW11], regression [CYXH12, ZWLY11] and classification [GHT14, MWF14].

In order to ensure that the optimal feature subset with respect to the goal concept has been found, a feature selection method would need to evaluate a total of $2^m - 1$ subsets, where m is the total number of features in the data set (an empty feature subset is excluded). This is computationally infeasible even for a moderately large m. For this reason, most newer feature selection approaches are filter methods which are computationally faster.

Uni-variate feature filters evaluate (and usually rank) a single feature, while multivariate filters evaluate an entire feature subset. Feature subset generation for multivariate filters depends on the search strategy. While there are many search strategies, there are four typical starting points for feature subset generation: (1) forward selection, (2) backward elimination, (3) bidirectional selection, and (4) heuristic feature subset selection. Forward selection typically starts with an empty feature set and then adds one or more features to the set. Backward elimination typically starts with the whole feature set and considers removing one or more features from the set. Bidirectional search searches from

both directions, simultaneously considering larger and smaller feature subsets. Heuristic selection generates a starting subset based on a heuristic (for example a genetic algorithm) and then explores it further.

In this chapter, a new filter-based feature selection method, with a heuristic feature elimination strategy based on redundancy is presented. Particularly, the proposed method has two stages: first it ranks the features in a univariate way, then reduces the number of features using a heuristic strategy which eliminates redundancy in the set of features. This can be thought of being a type of backward elimination. In this way, the method preserves all the advantage of filter methods, including the computational speed, while it reduces the chance to include redundant features, which is a drawback of uni-variate filter based methods. We demonstrate the effectiveness of our approach with extensive numerical experiments using credit data sets.

2.3 Related Work

Feature selection strategies based on filter methods have received attention from many researchers in statistics and machine learning. Their advantages are that they are fast, scalable and easy to interpret. The characteristics of filter based methods are as follows: *(i)* Redundant features may be included, *(ii)* Some features which as a group have strong discriminatory power but are weak as individual features will be ignored, and *(iii)* The filtering procedure is independent of the classifying method. The first 2 are considered negative, and the third can be thought of as being positive as it decreases the time it takes for the algorithm to find the final subset of features.

While univariate methods are generally faster, they are also less accurate compared with multivariate methods. However, these methods require a search strategy to determine the promising feature subset candidates. Many of these strategies are based on existing heuristics.

The conventional feature selection method with a genetic algorithm has difficulty for large-scale feature selection. The authors of [HC06] modify the representation of chromosome to be suitable for large-scale feature selection and adopt *speciation* to enhance the performance of feature selection by obtaining diverse solutions.

In [OLM04], hybrid genetic algorithms (GAs) are proposed that include local

search operators, which add (remove) the most (least) significant features to feature subsets. The authors of [CL10] compared their 4 different types of hybrid-SVM models; LDA+SVM, DT+SVM, F-Score+SVM and RST+SVM. The SVM part of their methods was for classification, while the other parts were for feature selection. Their results show that using the F-Score value on the German credit dataset was their best method, while on the Australian dataset, the linear discriminate analysis was their best approach.

In [HCX07] a hybrid genetic algorithm is used to find a subset of features that are most relevant to the classification task. Two stages of optimization are involved. The outer stage searches for the best subset of features via a wrapper method, in which the mutual information between the predictive labels of a trained classifier and the true classes serve as the fitness function for the genetic algorithm. The inner optimization performs local search via a filter method, in which an improved estimation of the conditional mutual information acts as an independent measure for feature ranking, taking account of both the relevance of the candidate feature to the output classes and redundancy with respect to the already-selected features.

The authors of [HGC08] focuses on enhancing the effectiveness of filter feature selection models from two aspects. First, a feature-searching engine is modified based on optimization theory. Second, a point injection strategy is designed to improve the regularization capability of feature selection. They apply these strategies to modify two typical filter models: SFS-based and GA-based approaches.

[HCW07] introduced a SVM classifier combined with a genetic algorithm (SVM+GA). Their approach simultaneously carried out feature selection while using a grid search to optimise the SVM parameters. The authors compared the SVM+GA to their other method of using the features F-Score to select the input features instead of the genetic algorithm. They show that the SVM+GA achieves better classification accuracy on 2 credit datasets, while also selecting fewer features than using F-Score.

In [AA05] an Ant Colony Optimization approach was presented for feature selection problems. The author calculates a term called "updated selection measure" which is used for selecting features, a function of the pheromone trail and the so called "local importance" which replaces the heuristic function.

The authors of [WYT+07] proposed a feature selection strategy based on rough

sets and particle swarm optimization. Rough sets have been used as a feature selection method with much success, but current hill-climbing rough set approaches to feature selection are inadequate at finding optimal reductions, as no heuristic can guarantee optimality.

[LO10] introduced a method for finding a minimum subset of features using an approximation of the Markov Blanket. The authors' version uses Hilbert-Schmidt Independence criterion, instead of the standard conditional test, as the measure of dependence between features in a kernel-induced space. This measurement allows the algorithm to remove both redundancy and irrelevance in the dataset at the same time.

[OGO+01] combined, information theory to select the relevant features, with neural networks for the classification accuracy. [Wes00] also experimented with neural networks as the classifier for credit scoring, and showed that without a feature selection process before classification, the accuracy on both the German and Australian credit data are not as accurate. [OHT05] criticized the use of neural networks in this area due to poor performance and for retaining irrelevant features, and their results show that a genetic programming approach outperformed a multi-layer perceptron.

[HS99] used a correlation based filter approach, which used a forward selection and backward elimination search method in-conjunction with a standardized Pearson's correlation to evaluate the goodness of the subsets. To measure the correlation between features and feature to target, the authors calculated the uncertainty coefficient [PTVF88], and used an instance based learner, Naive Bayes and C4.5 decision tree as the classifier for the classification accuracy of the chosen subsets.

[JW16] proposed method measured the correlation between continuous and discrete features by maximizing the sum of squares within groups while at the same time minimizing the sum of squares between groups. Their approach, called ECMBF based on a pair-wise correlation analysis, removed irrelevant features due to the correlation between features and target.

Table 2.1: Credit datasets

Data set	Size	Continuous / Nominal Features	Train / Test
German	1000	17 / 3	700 / 300
German (numeric)	1000	24 / 0	700 / 300
Australian	690	8 / 6	483 / 207
Japanese	684	9 / 6	479 / 205

2.4 Classifiers and Datasets

2.4.1 Credit Scoring Datasets

The credit scoring datasets were initially chosen as the original plan for the thesis was fraud detection on insurance data. Therefore, since credit scoring data would be of a similar type, it was an appropriate choice.

RRD was evaluated on 4 credit data sets from the UCI Machine Learning Repository [AN07a], each with a binary target variable. We decided to evaluate our method thoroughly on one type of data first, and then carry out evaluation experiments on 2 very different types of data to check its robustness performance. Feature selection for credit data has been the subject of several recent papers [CL10, OGO$^+$01, HCW07].

Table 1 provides an overview of the 4 data sets used in the numerical experiments. The German data set has 3 continuous features, 4 ordinal features and 13 nominal features, while the numerical version of the German data set has 24 continuous features. Both the Australian and Japanese data sets have 6 continuous features, the Australian data set also has 8 nominal features, and the Japanese data set has 9 nominal features. These are popular data sets for evaluating classification and feature selection methods, especially where credit scoring is the research topic.

2.4.2 Discretization

Our method assumes that we can compute a statistic $s(x, y)$ for all features and/or target variables x and y. Thus before we can apply the method, if the target and/or features have mixed types (numerical, ordinal, nominal) they must be first pre-processed so that they are all of the same type.

In classification problems we are typically faced with mixed-type features and a nominal (often binary) target, so in this chapter we transform all data to

nominal form via discretization. Machine learning algorithms such as Naive Bayes and Random Forests also perform better when the training and testing data are discrete.

It should be noted that if the data set being analysed has all continuous numerical features and either a binary or continuous target variable, RRD can use statistics such as Spearman's correlation to reduce the number of features. This removes the need for discretization, which simplifies RRD and improves its computational efficiency.

Discretization (or binning) groups continuous features into categories defined by specific range', thus converting them to nominal features. Discretization can be either supervised (using Chi-squared discretization [FHT01]) or unsupervised (using k equal-width bins [FHT01]). There are many suggested ways for choosing k, and we use the Freedman-Diaconis rule which is robust to outliers [FD81]:

$$k = \frac{(\max_i - \min_i)}{h} \quad \text{where} \quad h = \frac{2 \times IQR}{\sqrt[3]{n}}$$

and max_i, min_i, IQR and n are the maximum, minimum, inter-quartile range and number of the features, respectively.

We carry out experiments using the following discretization methods to explore the effect of the different binning, and to determine the one best suited to credit data:

- ***Equal width interval*** is one of the most commonly used binning methods, due to its simplicity and good results an a large range of problems. The algorithm divides the range of a features into k intervals of equal width, determined by:

  $$W = \frac{max - min}{k}$$

  This algorithm can be prone to outliers within the features, thus, seriously skewing the range. Using the Freedman-Diaconis rule, stated above, can help counteract this issue.

- ***Equal frequency interval*** is similar to equal width binning, above, but instead, it divides the features into k groups containing approximately equal number of samples.

- ***k-means*** is a clustering technique to bin the continuous values into k bins. The k-means clustering aims to minimizes the sum of squares between the samples and their assigned cluster centre. It iteratively adjusts which

samples belong to which cluster until the optimization problem converges. Although this technique finds natural occurring clusters in features, it can be computationally expensive and there is a possibility that small distinct clusters get swallowed up by another cluster, or if the number of k bins is too great large clusters will be divided into smaller similar clusters.

- ***Hierarchical clustering*** discretization: we use a *divisive* approach in which values start as one cluster, which is split using a measure of dissimilarity between pairs of observations as we move down the hierarchy.
- ***Jenks natural breaks optimization*** discretization groups the data into k bins in an iterative process, by minimizing within-cluster variance and maximizing between-cluster distance. This is similar to Fisher discretization but achieves greater separation between bins.
- ***Fisher*** discretization clusters the data in the feature into k bins using the "exact optimization" method proposed by W. D. Fisher which maximizes the between-cluster distance (measured as the sum of squares between centroids). This means that it clusters the data in the feature and maximizes the distance between each cluster (bin) using a sum of squares.
- ***ChiMerge*** is a supervised way of discretizing the features. The method is initialized by putting each sample into its own bin and then by using the Chi^2 statistic,

$$\tilde{\chi}^2 = \frac{1}{d}\sum_{k=1}^{n}\frac{(O_k - E_k)^2}{E_k}$$

  it is determined when adjacent intervals can be merged. Chi^2 empirically calculates the expected frequency (E_k) compared with the observed (O_k), and tests that the two intervals are statistically independent. By using a user-defined Chi^2 threshold, which sets the maximum value of Chi^2 that allows intervals to merge, the number of bins created is controlled.
- ***Minimum Description Length Principle*** was the final discretization method explored and was also a supervised technique. It uses entropy as a measure to calculate the variance within the bin and a Minimum Description Length to select the number of intervals to use.

In unsupervised discretization, the continuous feature is binned without regard to the target variable, and therefore it is a naive but fast binning method, although k-means binning can be computational costly. Whereas, in supervised binning the intervals are selected to optimize the correlation between the fea-

ture and target variable. This way of binning, in general, can improve classification predictions but is very expensive, and the improvements might not warrant the cost. In this research, when performing discretization, we ensured our methods had no bias or leakage (the test and validation data is kept completely separate from the training data, ensuring neither dataset had an influence on the binning decision) by only determining the cut-points for the intervals on the training data set, and used that information on the validation data set and test data set.

Optimal subsets were created as suggested by both the statistical metrics and the redundancy method. These subsets were evaluated using three popular machine learning algorithms and stratified 100-fold Monte Carlo cross-validation based on 70%/30% training/testing splitting of the data (see section 2.5.1). Mean accuracy was computed and is shown below, along with the number of selected features in each case.

2.4.3 Statistics

We experimented with several statistics:

- ***Pearson's*** χ^2 (Eqn. 2.1) is a measure of association between nominal features. The closer the chi-squared value is to 1, the more information is contained in one feature compared to another, while a value of 0 means that the features are independent of each other [ZS15].

$$\chi^2 = \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k} \tag{2.1}$$

  where:
  χ^2 is derived from Pearson's chi-squared test,
  O_k is the total of observations of type k,
  E_k is the expected count of type k,
  n is the cell count in the table.

- ***Cramer's*** ϕ_c (Eqn. 2.2) is a Pearson's chi-squared based statistic which measures the strength of the association between nominal features. Its range is between 0 and 1, 0 being no association and 1 being complete association. This statistic has been used successfully for feature selection

[WZZ13].

$$\phi_c = \sqrt{\frac{\chi^2/N}{min(n-1, r-1)}} \tag{2.2}$$

where:
χ^2 is derived from Pearson's chi-squared test,
N is the total number of observations,
n being the number of features,
r being the sample count.

- ***Stuart's*** τ_c (Eqn. 2.3) measures the correlation between categorical features, and its value ranges from -1 for perfect negative association to 1 for perfect positive association, with 0 indicating no association [Agr03].

$$\tau_c = \frac{2(n_c - n_d)}{n^2 \frac{(m-1)}{m}} \tag{2.3}$$

where:
n_c is the number of concordant pairs (classification direction is the same),
n_d is the number of discordant pairs (classification direction is not the same),
r is the number of samples,
c is the number of features,
m is the minimum value between the number or features and samples.

- ***Tschuprow's*** T (Eqn. 2.4) calculates the correlation between categorical features with a range of -1 to 1 for 2 by 2 tables and 0 to 1 for all other size tables [TK39].

$$T = \sqrt{\frac{\chi^2/n}{\sqrt{(r-1)(c-1)}}} \tag{2.4}$$

where:
χ^2 is derived from Pearson's chi-squared test,
n is the total number of observations,
c being the number of features,
r being the sample count.

- ***Contingency Coefficient*** C (Eqn. 2.5) calculates the correlation between

categorical features with a range of 0 to 1. It is scale invariant meaning, the value of the coefficient is maintained once the values within the table change relative to each other [Agr03].

$$C = \sqrt{\frac{\chi^2}{N + \chi^2}} \tag{2.5}$$

where:
χ^2 is derived from Pearson's chi-squared test,
N is the total sample size.

- ***Goodman & Kruskal's*** λ (Eqn. 2.6) is a measure of proportional reduction in error between nominal features, similar to Cramer's ϕ_c, and its value ranges from 0 to 1 [GK79].

$$\lambda = \frac{\epsilon_1 - \epsilon_2}{\epsilon_1} \tag{2.6}$$

where:
ϵ_1 is a measurement of the unconditional variability in a variable,
ϵ_2 is the same measure of variability, but conditional on the target.

- ***Goodman & Kruskal's*** γ (Eqn. 2.7) is a metric used to measure the association between categorical data [GK79].

$$\gamma = \frac{N_s - N_d}{N_s + N_d} \tag{2.7}$$

where:
N_s is the number of concordant pairs,
N_d is the number of reversed pairs.

- ***Spearman Correlation*** ρ (Eqn. 2.8) is a metric used to measure the association between 2 features [Spe61]. ρ will always be between 1 (a perfect positive correlation) and -1 (a perfect negative correlation).

$$\rho = \frac{COV(X, Y)}{\sigma_X \sigma_Y} \tag{2.8}$$

where:

$COV(X, Y)$ is the covariance of the features,
σ_X and σ_Y are the standard deviations of the features.

- ***Somers'*** δ (Eqn. 2.9) is a metric used to measure the association between categorical data [GK79].

$$\delta = \frac{N_C - N_D}{n(n-1)/2} \tag{2.9}$$

where:
N_C is the number of concordant pairs,
N_D is the number of discordant pairs,
n is the total number of observations.

- ***Mutual Information*** I (Eqn. 2.10) measures how much information can be obtained on one feature from another, and it has been used frequently in research on feature selection [YL03a]. It is an information-theory based measure of the similarity between two features or a feature and the target. A mutual information score of zero indicates that the features are independent, while a higher score shows a greater reduction in the uncertainty between features. Let *X* be a random variable and $P_X(x)$ be its probability distribution, then its entropy, *H(X)*, is defined as:

$$H(X) = \sum_x P_X(x) \log P_X(x) \tag{2.10}$$

Next we define the conditional entropy, $H(X \mid Y)$ as:

$$H(X|Y) = \sum_y P_Y(y) \Big[- \sum_x P_{(X|Y)}(x \mid y) \log \Big(P_{(X|Y)}(x \mid y) \Big) \Big] \tag{2.11}$$

where $P_{(X|Y)}(x) = P_{XY}(x, y)/P_Y(y)$.

Finally, putting eqn. 2.10 and eqn. 2.11 together we get the definition for Mutual Information, $I(X; Y)$.

$$I(X; Y) = H(X) - H(X|Y) \tag{2.12}$$

- ***Theil's*** U (Eqn. 2.13) is the Uncertainty Coefficient and is a measure of the proportion of uncertainty within one feature that is explained by another [The72]. It is based on the idea of information entropy.

$$Theil's\, U = \frac{H(X) - H(X|Y)}{H(X)} \tag{2.13}$$

where:

$H(X)$ is the entropy of a single feature, which is a measure of its uncertainty (Eqn. 2.10),
$H(X|Y)$ is the conditional entropy between features or feature to target, which is a measure of a feature's uncertainty given another variable (Eqn. 2.11).

- ***Messenger & Mandell's*** Θ (Eqn. 2.14), can use table 2.2, but without taking the class weights into account. It was developed for use in decision trees (known as THAID) before being replace with Chi-squared AID (CHAID) [MM72]. The statistic also ranges from 0 to 1, with 0 indicating no association between features and 1 that the features are completely dependent.

$$\Theta = \frac{\max(a,b) + \max(c,d) + \max(e,f)}{a+b+c+d+e+f}. \tag{2.14}$$

- Θ^* (Eqn. 2.15), our proposed new statistic, is based on the original Messenger and Mandell Θ [MM72] but takes the class frequency into account when calculating the value. By doing this, it helps with unbalanced data sets, such as the German credit data. To calculate the Θ^* correlation between a feature and the target variable given a contingency table, as shown in table 2.2, with a binary target variable in the ratio of m:n, we use the following (which by taking the class imbalance into account helps to achieve a fairer balanced statistic);

Table 2.2: Contingency table of Feature and Target Variable

	Target Variable	
Feature Levels	*Class 1*	*Class 2*
1	a	b
2	c	d
3	e	f

$$b \times \frac{m}{n} = b^*; d \times \frac{m}{n} = d^*; f \times \frac{m}{n} = f^*$$
$$\Theta^* = \frac{\max(a, b^*) + \max(c, d^*) + \max(e, f^*)}{a + b^* + c + d^* + e + f^*}. \tag{2.15}$$

2.4.4 Classifiers

We used 3 classifiers: Logistic Regression, Random Forests and Naive Bayes. We chose these three methods because they are popular yet very different, and are widely used in the credit-score research area. All three can handle mixed data types, but often perform better when continuous features are transformed into nominal form [DKS95].

- ***Logistic Regression***, shown in Fig.2.1, is widely used for binary classification problems, modeling the probabilities of the 2 possible outcomes. It is an extension of the linear regression algorithm, making it popular due to its simplicity while achieving competitive results. In binary classification, we are determining the probability of a sample belonging to one of two classes, usually defined as class 0 and class 1. Therefore, since having a probability between $0-1$, the Logistic (Sigmoid) function (equation 2.16) is ideal.

$$f(x) = \frac{1}{1 + e^{-z}} \tag{2.16}$$

  The z in the Sigmoid equation is the output from Linear Regression, which is squashed between $0 - 1$. The advantage of using the Sigmoid equation is that its derivative is easy to calculate. Then, by setting a threshold, usually the midpoint 0.5, we can classify the predicted values into class 0

or class 1.

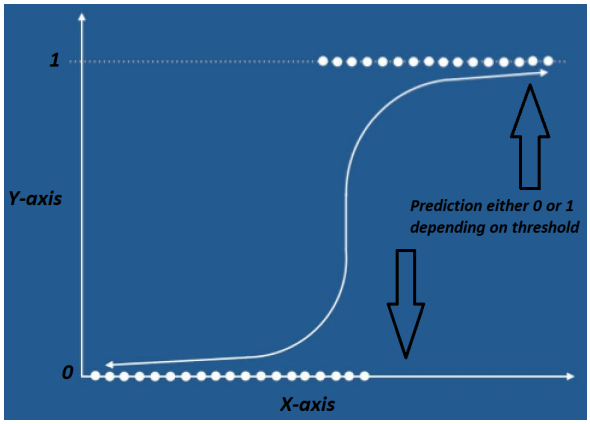

Figure 2.1: A Logistic Regression graph showing how it uses a sigmoid function for its binary classification predictions.

- ***Random Forests*** [Bre17] is an ensemble learning algorithm, created by a number of decision trees pre-determined by the user. Each decision tree is trained on a subset of the training data. Both features and observations are randomly sampled to create the subset. A decision tree can be thought as a sequence of if/else conditions, which flows out to a predicted class based on the majority vote system. The trained model is applied to the test set, and its final classification decision is the one where most trees have voted for it. An individual decision tree is very explainable, but the random forest algorithm can become complicated to explain due to the many decision trees used to create it. Random forests have achieved state-of-the-art results in many areas; due to their ensemble structure, it makes a wrong classification only when more than half the decision trees are wrong. It also lowers the possibility of over-fitting by averaging all the decision trees. Random forests are a lot more complex and computationally cost more than decision tress, and other classifiers. Fig. 2.2 shows how a random forest used majority voting to determine Class B, yellow nodes are selected. The sample is passed through n decision trees, and based on conditions, at the final nodes the selected classes are highlighted using a red square.
- ***Naive Bayes*** is a probabilistic algorithm, based on Bayes' Theorem, that is typically used for classification problems. Naive Bayes is regarded as a simple, intuitive classifier, that performs well in many research areas. It has 2 main assumptions; it is assumed that all features are independent, and that all features contribute equally to the prediction, with none

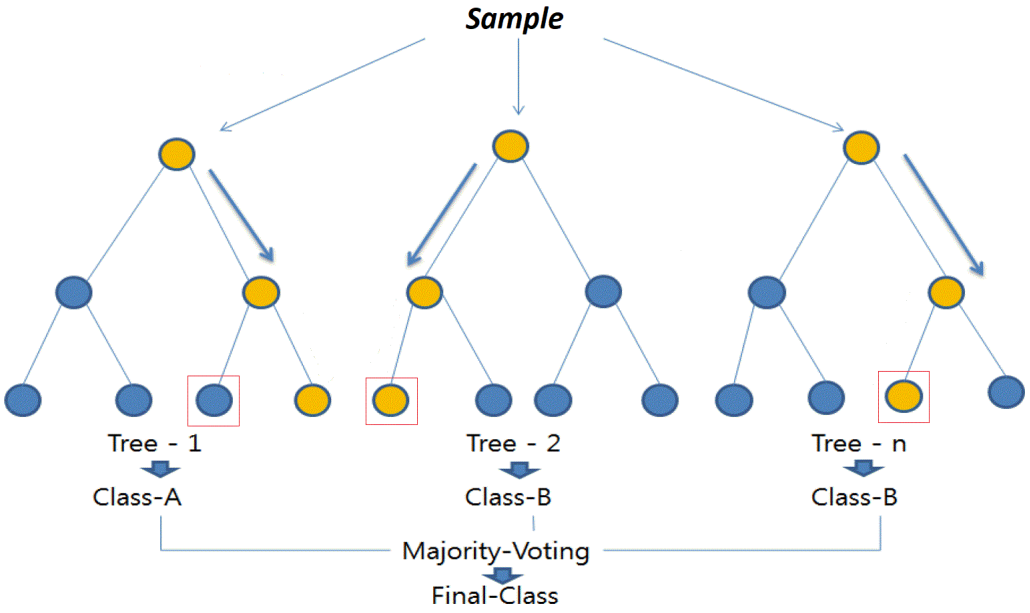

Figure 2.2: Random forest example, with n decision trees, predicting the Class B (yellow node) through majority voting.

being irrelevant. It should be noted that in real-world data sets, the independence assumption is rarely respected, but the classifier still works well in practice. The crux of the Naive Bayes classifier is based on Bayes Theorem, as shown in equation 2.17. Naive Bayes classifier is a very fast algorithm, making decisions in real-time.

$$P(\mathbf{A}|\mathbf{B}) = \frac{P(\mathbf{B}|\mathbf{A})P(\mathbf{A})}{P(\mathbf{B})} \tag{2.17}$$

2.5 Materials and Methods

2.5.1 Experimental Design

We used stratified 100-fold Monte Carlo cross-validation with $70\%/30\%$ training/testing splitting of the data, as follows: (*i*) randomly split the data set samples into 70% training set and 30% testing set; (*ii*) using only the training set, create bins for the discretization method; (*iii*) run RRD on the discretized training set to find the optimal features; (*iv*) using the training set subset, build the predictor model (e.g. Naive Bayes); (*v*) select the same features in the test set as RRD selected in the training set; (*vi*) using the bin cut-points found in step (*ii*) discretize the test set; (*vii*) evaluate the predictor model built in step (*iv*) using the test set, which has been kept completely isolated from the training

set, and finally (*viii*) repeat steps (*i*) to (*vii*) k times (in this part of the research $k = 100$). Average prediction accuracy and number of selected features, along with their corresponding standard deviations, are reported in each case.

Monte Carlo cross-validation randomly samples the data, splitting it into different subsets for training set and test set, for each k in the cross-validation. The training data is randomly split further into a smaller training set and a validation set, which is set aside, and the model is trained only on the training section of the set. Feature selection using *Relevance-Redundancy Dominance*, and any hyper-parameter tuning for the model is carried out on the training set and evaluated on the validation set. And finally, the classification accuracy is averaged over the k number of iterations within the cross-validation using the test set. An advantage of this method is the split size is not determined by the k-fold, but there is a chance that some of the observations in the data may never be selected in the test set, hence we will not know how the model classifies them. Fig. 2.3 shows how random samples are selected from within the data set to be part of the test set. Unlike a 5-fold or 10-fold cross-validation, there is no pattern to the selecting process.

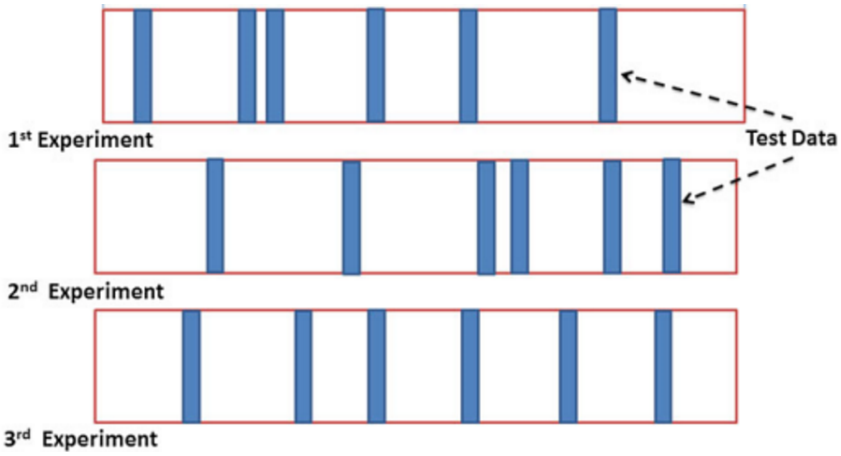

Figure 2.3: An example of a Monte Carlo Cross-Validation on 3 folds in the data set. The diagram only shows the test dataset in blue, the training and validation datasets are the remaining white sections left over.

A stratified approach is used to generate the random sample splits, which means the 3 sets maintain the class ratio as the full data set, thus a true representative sample of the population. Fig. 2.4 shows how a 2 class data set, where the classes are imbalanced similar to the German credit data set, are split in a stratified approach for a 5-fold cross-validation.

By performing 100-fold cross-validation, it reduces the variance in the mean

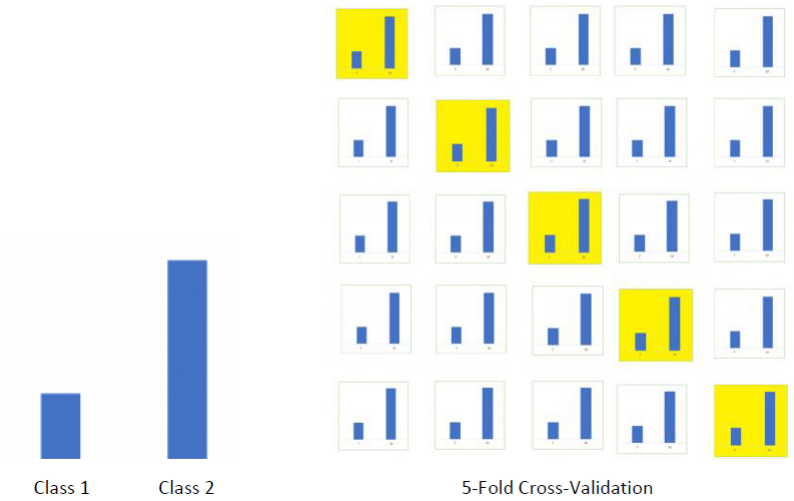

Figure 2.4: An example of a Stratified Cross-Validation where the data is imbalanced. As shown on the right, class 2 has more samples than class 1, and when split into 5 fold cross-validation, the stratified approach keeps the imbalance between the classes in each fold.

cross-validation accuracy result, therefore our stated results below are more stable than other methods presented using a smaller value of k for the k-fold. But by selecting a k much less than the number of samples, and therefore much less than Leave-One-Out cross-validation, we ensure our results also have a low bias. If k was set equal to the number of samples, the error estimates would be very low in bias but with the possibility of a high variance, while if k was set to a low value $(5,\ 10)$, the variance would be low but with high bias. This is why we selected k equal to 100, as it was a good k value for the bias-variance tradeoff, which is very important for model evaluation.

2.5.2 Relevance-Redundancy Dominance

Relevance-Redundancy Dominance (RRD) is a univariate filter-based feature selection method, which can use any suitable statistic to select a *good* subset of features in a data set. The statistic can be symmetric ($s(f, f') \equiv s(f', f)$ for example correlation or mutual information) or asymmetric (for example Goodman and Kruskal's λ [GK79]).

The idea behind RRD is to remove features from the data which are explained by other more relevant features. By ordering the features of the dataset to their relevance to the target, we ensure to keep the important features. Next, we say a feature is redundant, due to be dominated, when a less important (weaker) feature is explained more by a dominant feature than how much the

weaker feature explains the target. The motivation behind this idea is completely novel, but works very well as the results show, by combining relevancy and redundancy in a unique way.

Given a binary statistic s, features $f \in F$ and a target variable t, the RRD method works as follows. As in other methods, the features are ranked for relevance using s: f is more relevant than f' if $s(f,t) > s(f',t)$. We shall say that f *dominates* f' if $s(f,t) > s(f',t)$ and $s(f,f') > s(f',t)$. Although the statistics used can be symmetric, it is important that when they are asymmetric, the order of the variables are kept as above, as we are only interested when f' is being dominated. We shall also say that f' is *redundant* if it is dominated by f and f is not dominated by any other feature, as if f was dominated it should not have been retained in the final subset of features. RRD selects all non-redundant features.

This leads to the feature selection method shown in Algorithm 1. First we pre-compute the statistics $x_{ft} = s(f,t)$ between each feature f and the target variable t, and initialise the set of selected features to the empty set $\emptyset$. Then we select the feature $\hat{f} \in F$ with greatest relevance $x_{\hat{f}t}$, generate the set R of $f \in F$ that are made redundant by $\hat{f}$, add $\hat{f}$ to S, and remove $R \cup \{\hat{f}\}$ from F. The last few steps are repeated until F is empty, then we return the set S of selected features.

Algorithm 1 RRD Feature Selection Algorithm

1: given features F and target t
2: **for all** $f \in F$ **do**
3: $\quad x_{ft} \leftarrow s(f,t)$
4: **end for**
5: $S \leftarrow \emptyset$
6: **while** $F \neq \emptyset$ **do**
7: $\quad \hat{f} \leftarrow \arg\max_{f \in F} x_{ft}$
8: $\quad R \leftarrow \{f \mid f \in F \wedge s(\hat{f},f) > x_{ft}\}$
9: $\quad F \leftarrow F \setminus (R \cup \{\hat{f}\})$
10: $\quad S \leftarrow S \cup \{\hat{f}\}$
11: **end while**
12: **return** S

The following bullet points give a line-by-line description of Algorithm 1:

- In line 1 we define the features F and the target t from the dataset.
- Lines $2-4$ are a loop; this is exited (line 4) when all the features of the dataset have been analyzed.

- The purpose of this loop is to calculate a given statistic between each feature and the target.
- Line 5 is where we initialize S as an empty set, which we will fill with the selected features.
- Lines $6-11$ are a loop; this is exited (line 11) when the set of features F contains no more features f.
- This loop selects the important features, and removes the dominated features.
- Line 7 selects the feature ($\hat{f}$) which has the greatest relevance, with respect to the target.
- Line 8 creates a set of features R, which are dominated by the current most relevant feature ($\hat{f}$).
- Line 9 is where the set of dominated features R, and the current most relevant feature ($\hat{f}$) are removed from the set of features F.
- In line 10 we insert the current most relevant feature ($\hat{f}$) into the pre-initialized set S.
- Line 12 ends the algorithm and returns the selected set of feature S.

Note that Algorithm 1 typically does not compute all s-values between features (for example a full correlation matrix). This is because after a feature has been removed from F no further statistics on it need be computed. This means that it will often compute fewer statistics than (say) the MRMR method of [PLD05a], though in the worst case it computes the same number. It is possible to construct data sets for which RRD removes no features at all, or removes all but one, but in practice we find that it usually generates a small subset of features.

Finally, it should be pointed out that RRD assumes that we can compute statistic $s(x, y)$ for all $x, y \in F \cup \{t\}$. Thus before we can apply RRD, if the target and/or features have mixed types (numerical, ordinal, nominal) they must be first pre-processed so that they are all of the same type. This pre-processing was detailed above in the discretization section 2.4.2.

Table 2.3: German credit data set RRD results

Classifier	Statistic	Selected	Accuracy	Discretization
Logistic Model	Messenger & Mandell's Θ	13.27 ± 0.45	75.32 ± 2.12	Equal Width
	Θ^*	10.04 ± 1.16	74.76 ± 2.33	Equal Width
	Cramer's ϕ_c	5.18 ± 0.78	74.64 ± 2.33	k-means
	Stuart's τ_c	5.19 ± 0.9	73.92 ± 2.02	k-means
	Tschuprow's T	3.59 ± 0.55	73.7 ± 2.1	k-means
	Theil's U	3.61 ± 0.75	73.58 ± 2.11	k-means
	Mutual Information I	2.17 ± 0.38	72.73 ± 2.06	k-means
	Goodman & Kruskal's γ	5.05 ± 0.9	72.69 ± 1.84	Chi^2
	Contingency Coefficient C	2.26 ± 0.52	72.62 ± 1.9	k-means
	Pearson's χ^2	2.24 ± 0.51	72.62 ± 1.89	k-means
	Somers' δ	4.32 ± 0.74	71.69 ± 2.14	Chi^2
	Goodman & Kruskal's λ	1.12 ± 0.33	71.6 ± 1.44	Equal Width
Naive Bayes	Messenger & Mandell's Θ	13.27 ± 0.45	75.66 ± 2.32	k-means
	Cramer's ϕ_c	5.22 ± 0.73	74.44 ± 2.1	k-means
	Stuart's τ_c	5.18 ± 0.94	74.31 ± 2.27	k-means
	Θ^*	12.3 ± 0.18	74.15 ± 1.71	Equal Width
	Tschuprow's T	3.6 ± 0.64	73.74 ± 2.16	k-means
	Theil's U	3.56 ± 0.74	73.65 ± 2.28	k-means
	Mutual Information I	2.21 ± 0.43	72.54 ± 2.38	k-means
	Pearson's χ^2	2.2 ± 0.43	72.49 ± 2.3	k-means
	Contingency Coefficient C	2.24 ± 0.45	72.46 ± 2.33	k-means
	Goodman & Kruskal's γ	5.09 ± 1.04	71.99 ± 2.34	Chi^2
	Somers' δ	4.2 ± 0.71	71.67 ± 2.49	Chi^2
	Goodman & Kruskal's λ	2.07 ± 0.36	70.99 ± 1.67	k-means
Random Forest	Messenger & Mandell's Θ	13.3 ± 0.46	76.13 ± 1.76	k-means
	Θ^*	10.85 ± 0.84	75.36 ± 2.37	k-means
	Cramer's ϕ_c	5.12 ± 0.74	73.74 ± 2.35	k-means
	Theil's U	3.55 ± 0.73	73.22 ± 2.39	k-means
	Tschuprow's T	3.51 ± 0.56	73.18 ± 2.29	k-means
	Stuart's τ_c	5.16 ± 0.88	72.56 ± 2.43	k-means
	Mutual Information I	2.22 ± 0.44	72.07 ± 1.8	k-means
	Pearson's χ^2	2.23 ± 0.49	71.97 ± 1.7	k-means
	Contingency Coefficient C	2.27 ± 0.51	71.95 ± 1.71	k-means
	Somers' δ	4.77 ± 0.79	71.7 ± 2.29	Chi^2
	Goodman & Kruskal's λ	2.08 ± 0.34	71.66 ± 1.44	k-means
	Goodman & Kruskal's γ	5.05 ± 1.08	71.65 ± 2.07	Chi^2

Table 2.4: German numerical credit data set RRD results

Classifier	Statistic	Selected	Accuracy	Discretization
Logistic Model	Θ^*	9.73 ± 1.12	75.73 ± 0.59	Equal Width
	Messenger & Mandell's Θ	12 ± 0	75.15 ± 1.85	Equal Width
	Cramer's ϕ_c	4.86 ± 0.73	74.36 ± 1.89	Equal Width
	Stuart's τ_c	5.72 ± 0.92	74.14 ± 1.87	Equal Width
	Goodman & Kruskal's γ	5.85 ± 0.74	74.04 ± 1.83	k-means
	Spearman Correlation ρ	5.19 ± 0.93	74.04 ± 1.84	NONE
	Theil's U	3.78 ± 0.73	73.93 ± 1.9	Equal Width
	Tschuprow's T	5.07 ± 0.79	73.67 ± 1.78	Equal Width
	Contingency Coefficient C	5.21 ± 0.88	73.05 ± 1.44	Equal Width
	Pearson's χ^2	5.05 ± 0.93	73.01 ± 1.46	Equal Width
	Mutual Information I	5.03 ± 0.92	72.99 ± 1.5	Equal Width
	Somers' δ	7.04 ± 1.17	72.88 ± 2.27	Chi^2
	Goodman & Kruskal's λ	2.08 ± 0.34	71.62 ± 1.39	Equal Width
Naive Bayes	Θ^*	10 ± 0	75.33 ± 1.38	Equal Width
	Messenger & Mandell's Θ	11.98 ± 0.14	75.28 ± 2.05	Equal Width
	Cramer's ϕ_c	4.9 ± 0.76	73.53 ± 2.13	Equal Width
	Theil's U	3.8 ± 0.8	73.02 ± 2.02	Equal Width
	Stuart's τ_c	5.69 ± 1.19	72.77 ± 2.29	Equal Width
	Spearman Correlation ρ	5.48 ± 0.95	72.69 ± 2.21	NONE
	Tschuprow's T	3.79 ± 0.73	72.37 ± 2.29	k-means
	Mutual Information I	3.91 ± 0.95	71.98 ± 2.4	Chi^2
	Pearson's χ^2	3.83 ± 0.97	71.88 ± 2.58	Chi^2
	Contingency Coefficient C	3.13 ± 0.9	71.87 ± 2.41	Equal Freq
	Goodman & Kruskal's γ	5.64 ± 0.73	71.86 ± 2.97	k-means
	Goodman & Kruskal's λ	2.1 ± 0.33	70.92 ± 1.79	Equal Width
	Somers' δ	4.47 ± 0.72	70.32 ± 3.34	k-means
Random Forest	Θ^*	10.21 ± 0.74	75.83 ± 0.78	Equal Width
	Messenger & Mandell's Θ	11.99 ± 0.1	75.38 ± 1.69	Equal Width
	Stuart's τ_c	4.91 ± 0.64	73.35 ± 2.19	k-means
	Pearson's χ^2	5.08 ± 0.8	73.2 ± 1.57	Equal Width
	Mutual Information I	5.12 ± 0.81	73.19 ± 1.65	Equal Width
	Cramer's ϕ_c	5.35 ± 0.78	73.17 ± 1.72	Equal Width
	Contingency Coefficient C	5.3 ± 0.82	73.13 ± 1.56	Equal Width
	Goodman & Kruskal's γ	5.71 ± 0.78	73.03 ± 1.97	k-means
	Tschuprow's T	5.8 ± 0.89	73.02 ± 1.73	Equal Width
	Theil's U	3.97 ± 0.82	72.97 ± 2.13	Equal Width
	Spearman Correlation ρ	5.25 ± 0.93	72.58 ± 2.25	NONE
	Somers' δ	6.97 ± 1.04	72.13 ± 2.07	Chi^2
	Goodman & Kruskal's λ	1.97 ± 0.17	71.76 ± 1.51	k-means

Table 2.5: Australian credit data set RRD results

Classifier	Statistic	Selected	Accuracy	Discretization
Logistic Model	Goodman & Kruskal's λ	7.62 ± 0.92	85.98 ± 2.14	Chi^2
	Tschuprow's T	6 ± 0.79	85.86 ± 2.07	Chi^2
	Cramer's ϕ_c	5.62 ± 0.81	85.83 ± 2.09	Chi^2
	Θ^*	7.92 ± 1.33	85.81 ± 2.37	Chi^2
	Goodman & Kruskal's γ	5.32 ± 0.74	85.78 ± 1.99	k-means
	Stuart's τ_c	5.33 ± 0.55	85.78 ± 1.78	Equal Freq
	Messenger & Mandell's Θ	8 ± 0	85.74 ± 1.87	Chi^2
	Theil's U	4.88 ± 0.79	85.54 ± 2.01	Chi^2
	Somers' δ	6.08 ± 0.84	85.37 ± 1.97	Chi^2
	Mutual Information I	4.28 ± 0.9	85.29 ± 1.88	Chi^2
	Pearson's χ^2	4 ± 0.85	85.27 ± 1.91	Chi^2
	Contingency Coefficient C	4.07 ± 0.84	85.26 ± 1.9	Chi^2
Naive Bayes	Somers' δ	3.94 ± 0.28	85.9 ± 2.19	Equal Freq
	Tschuprow's T	5.38 ± 0.68	85.53 ± 2.46	Equal Freq
	Messenger & Mandell's Θ	5.4 ± 0.49	85.47 ± 2.68	Equal Width
	Θ^*	8.03 ± 0.98	85.45 ± 1.65	Equal Freq
	Cramer's ϕ_c	6.21 ± 0.73	85.34 ± 2.62	Equal Freq
	Goodman & Kruskal's γ	4.99 ± 0.54	85.33 ± 2.3	Equal Freq
	Theil's U	5.45 ± 0.66	85.31 ± 2.46	Equal Freq
	Goodman & Kruskal's λ	7.91 ± 0.6	85.08 ± 2.21	Equal Freq
	Stuart's τ_c	5.24 ± 0.57	84.98 ± 2.47	Equal Freq
	Contingency Coefficient C	3.05 ± 0.46	84.63 ± 2.53	Equal Freq
	Pearson's χ^2	3.05 ± 0.46	84.63 ± 2.53	Equal Freq
	Mutual Information I	3.28 ± 0.71	84.54 ± 2.53	Equal Freq
Random Forest	Goodman & Kruskal's λ	8.58 ± 0.55	86.29 ± 1.97	Chi^2
	Θ^*	7.89 ± 0.47	86.17 ± 1.18	Chi^2
	Messenger & Mandell's Θ	8.05 ± 0.22	86.09 ± 1.87	k-means
	Cramer's ϕ_c	7.4 ± 0.55	86.06 ± 1.83	Equal Width
	Somers' δ	6.09 ± 0.87	85.71 ± 2.09	Chi^2
	Stuart's τ_c	6.49 ± 0.8	85.69 ± 1.92	k-means
	Tschuprow's T	5.2 ± 0.71	85.57 ± 2.05	Equal Width
	Goodman & Kruskal's γ	5.63 ± 0.77	85.49 ± 2.1	k-means
	Contingency Coefficient C	2.04 ± 0.2	85.34 ± 2.15	k-means
	Pearson's χ^2	2.04 ± 0.2	85.34 ± 2.15	k-means
	Mutual Information I	2.13 ± 0.34	85.33 ± 2.15	k-means
	Theil's U	2.97 ± 0.36	85.06 ± 2.13	Equal Width

Table 2.6: Japanese credit data set RRD results

Classifier	Statistic	Selected	Accuracy	Discretization
Logistic Model	Goodman & Kruskal's λ	7.43 $\pm$ 0.9	86.56 $\pm$ 2.07	Chi^2
	Stuart's τ_c	5.79 $\pm$ 0.82	86.55 $\pm$ 2.12	Chi^2
	Θ^*	9.3 $\pm$ 1.26	86.51 $\pm$ 2.03	Chi^2
	Tschuprow's T	6.17 $\pm$ 0.88	86.44 $\pm$ 2.1	Chi^2
	Cramer's ϕ_c	6.28 $\pm$ 0.87	86.42 $\pm$ 2.23	Equal Freq
	Goodman & Kruskal's γ	5.43 $\pm$ 0.98	86.31 $\pm$ 2.01	Chi^2
	Messenger & Mandell's Θ	7.98 $\pm$ 0.14	86.24 $\pm$ 2.25	Chi^2
	Theil's U	5.29 $\pm$ 0.91	86.24 $\pm$ 2.19	Chi^2
	Mutual Information I	4.64 $\pm$ 0.96	86.18 $\pm$ 2	Chi^2
	Somers' δ	6 $\pm$ 0.91	86.14 $\pm$ 2.05	Chi^2
	Contingency Coefficient C	4.53 $\pm$ 0.89	86.1 $\pm$ 2.06	Chi^2
	Pearson's χ^2	4.45 $\pm$ 0.91	86.1 $\pm$ 2.03	Chi^2
Naive Bayes	Somers' δ	3.95 $\pm$ 0.39	87.04 $\pm$ 1.94	Equal Freq
	Θ^*	8.66 $\pm$ 1.47	86.99 $\pm$ 0.09	Equal Width
	Cramer's ϕ_c	6.2 $\pm$ 0.77	86.47 $\pm$ 2.02	Equal Freq
	Messenger & Mandell's Θ	5.28 $\pm$ 0.45	86.45 $\pm$ 1.96	Equal Width
	Theil's U	5.52 $\pm$ 0.63	86.33 $\pm$ 2.09	Equal Freq
	Goodman & Kruskal's γ	4.65 $\pm$ 0.73	86.32 $\pm$ 1.89	Equal Freq
	Tschuprow's T	5.49 $\pm$ 0.88	86.22 $\pm$ 2.09	Equal Freq
	Stuart's τ_c	5.43 $\pm$ 0.86	86.12 $\pm$ 1.91	Equal Freq
	Goodman & Kruskal's λ	7.55 $\pm$ 0.61	85.86 $\pm$ 1.94	Equal Freq
	Contingency Coefficient C	3.03 $\pm$ 0.17	85.76 $\pm$ 1.94	Equal Width
	Pearson's χ^2	3.03 $\pm$ 0.17	85.71 $\pm$ 1.92	Equal Width
	Mutual Information I	3.06 $\pm$ 0.28	85.68 $\pm$ 2.18	Equal Width
Random Forest	Messenger & Mandell's Θ	8.13 $\pm$ 0.34	86.58 $\pm$ 2.14	k-means
	Θ^*	11.01 $\pm$ 2.3	86.57 $\pm$ 2.68	k-means
	Somers' δ	3.97 $\pm$ 0.3	86.52 $\pm$ 2.2	Chi^2
	Goodman & Kruskal's λ	8.16 $\pm$ 0.39	86.51 $\pm$ 1.97	Chi^2
	Tschuprow's T	6.19 $\pm$ 0.94	86.49 $\pm$ 2.22	Chi^2
	Cramer's ϕ_c	7.65 $\pm$ 0.48	86.41 $\pm$ 1.88	Chi^2
	Stuart's τ_c	5.97 $\pm$ 0.73	86.37 $\pm$ 2.18	k-means
	Goodman & Kruskal's γ	5.18 $\pm$ 0.99	86.3 $\pm$ 2.06	k-means
	Contingency Coefficient C	2.06 $\pm$ 0.24	86.13 $\pm$ 1.98	k-means
	Pearson's χ^2	2.06 $\pm$ 0.24	86.13 $\pm$ 1.98	k-means
	Mutual Information I	2.16 $\pm$ 0.37	86.12 $\pm$ 2.01	k-means
	Theil's U	4.57 $\pm$ 0.71	85.95 $\pm$ 1.85	k-means

Table 2.7: Feature selection results from related work

data set	selected	accuracy	algorithm	paper
German	6	76.21%	SBC	[RG02]
	6	76.13%	ABC	[RG02]
	7	75.85%	NN	[OGO+01]
	4	74.38%	NB ECMBF	[JW16]
	20	74.32%	NB Full-set	[JW16]
	14	73.87%	NB Consistency	[JW16]
	4	73.76%	NB FCBF	[JW16]
	3	73.24%	NB CFS	[JW16]
	4	72.18%	C4.5 ECMBF	[JW16]
	3	71.47%	C4.5 CFS	[JW16]
	4	71.32%	C4.5 FCBF	[JW16]
	14	71.26%	C4.5 Consistency	[JW16]
	20	71.26%	C4.5 Full-set	[JW16]
Japanese	2	85.45%	C4.5 ECMBF	[JW16]
	15	85.83%	C4.5 Full-set	[JW16]
	7	85.28%	C4.5 CFS	[JW16]
	2	84.94%	NB ECMBF	[JW16]
	6	84.77%	C4.5 FCBF	[JW16]
	13	84.68%	C4.5 Consistency	[JW16]
	15	78.13%	NB Full-set	[JW16]
	13	75.32%	NB Consistency	[JW16]
	6	75.23%	NB FCBF	[JW16]
	7	74.81%	NB CFS	[JW16]

2.6 Results

The best results for each classifier and statistic are shown in Tables 2.3, 2.4, 2.5 and 2.6, using various discretization methods. These can be compared to published results from various papers using both new and well established filter feature selection methods, shown in Tables 2.7 and 2.8. The proposed threshold-free method performed very well using various statistics, showing the methods adaptability to be used on all types of data.

The best RRD result on the German data set was 76.13% $\pm$ 1.76% with 13.3 $\pm$ 0.46 selected features, using Messenger & Mandell's Θ [MM72] and k-means discretization, using Random Forest as the classifier. It outperformed the majority of the other methods, and was on par with the best, the selective Bayesian classifier of [RG02] with 76.21% and 6 selected features. The worst result was 70.99% $\pm$ 1.67% with Goodman & Kruskal λ using Naive Bayes as the classifier and k-means discretization method. Θ^*, our proposed statistic, was only beaten

Table 2.8: Feature selection results from related work

data set	selected	accuracy	algorithm	paper
German	20.4	77.50%	SVM + Grid search+ F-Score	[HCW07]
(numerical)	12	76.70%	F-score + SVM	[CL10]
	12	76.10%	LDA + SVM	[CL10]
	12	75.60%	RST + SVM	[CL10]
	24	75.40%	Full-set + SVM	[CL10]
	12	73.70%	DT + SVM	[CL10]
Australian	7	86.52%	LDA + SVM	[CL10]
	7	86.29%	DT + SVM	[CL10]
	2	85.79%	C4.5 ECMBF	[JW16]
	2	85.79%	NB ECMBF	[JW16]
	7	85.22%	RST + SVM	[CL10]
	7	85.10%	F-score + SVM	[CL10]
	5	84.80%	C4.5 - LV F	[LS$^+$96]
	1	84.65%	IB1-CFS	[HS99]
	14	84.34%	Full-set + SVM	[CL10]
	7.6	84.20%	SVM + Grid search+ F-Score	[HCW07]
	14	83.91%	C4.5 Full-set	[JW16]
	13	83.83%	C4.5 Consistency	[JW16]
	1	83.78%	Naive-CFS	[HS99]
	7	83.49%	C4.5 FCBF	[JW16]
	7	83.32%	C4.5 CFS	[JW16]
	5	80.30%	ID3 - LV F	[LS$^+$96]
	14	76.09%	NB Full-set	[JW16]
	7	75.32%	NB CFS	[JW16]
	13	74.89%	NB Consistency	[JW16]
	7	73.57%	NB FCBF	[JW16]

by Messenger & Mandell's Θ [MM72] which it is based on using 2 classifiers, and also finished fourth with the Naive Bayes classifier.

On the numerical German data set our best result was 75.83% $\pm$ 0.78% using our proposed statistic Θ^* and equal width discretization, with 10.21 $\pm$ 0.74 selected features. This is beaten by the SVM + Grid search + F-score method of [HCW07] with 77.50% and approximately 20 selected features, so our accuracy is slightly worse but with 10 fewer features selected, which can give the user an option of reducing the optimal subset further without losing any noticeable accuracy. The worst result was 70.32% $\pm$ 3.34% with Somers' δ using Naive Bayes as the classifier and k-means discretization method. On this data set, our proposed statistic Θ^* was the best statistic using all 3 classifiers, confirming its competitiveness against the other tested statistics.

On the Australian data set Goodman & Kruskal's λ with Chi^2 binning and the Random Forest classifier was the most accurate with 86.29% $\pm$ 1.97%. This was better than the result from Chen [CL10] using DT + SVM. The worst result was 84.54% $\pm$ 2.53% with Mutual Information I using Naive Bayes as the classifier and equal frequency discretization method. The new proposed statistic Θ^* achieved second place using the Random Forest classifier, while obtaining fourth using the other 2 classifiers.

On the Japanese data set Somers' δ with equal frequency discretization was the most accurate: 87.04% $\pm$ 1.94% and 3.95 $\pm$ 0.39 selected features. This beats the ECMBF method of Jiang [JW16] with 2 selected features and 85.45%. The worst result was 85.68% $\pm$ 2.18% with Mutual Information I, Naive Bayes and equal width discretization. Our proposed statistic Θ^* scored in the top 3 of each of the classifiers, showing its robustness.

In summary, the best results were found by Random Forests, the most robust statistics were Messenger & Mandell's Θ and Cramer's ϕ_c, and the best binning methods were generally the unsupervised k-means and equal width discretization. The new statistic Θ^*, achieved the best results in the analysis of some of the data sets, showing it could be considered to be a good statistic to use in future for feature selection algorithms. Overall RRD performed excellently, selecting 6%– 65% of features while achieving very competitive accuracies compared to published results.

2.7 Further Experiments

All our experiments so far used credit scoring datasets. RRD performed very well on this type of data, therefore to evaluate it further, we tested it on two very different types of data: pharmaceutical data and microarray data. These datasets contained more features, thus a greater challenge to RRD and would also test its scalability.

The pharmaceutical *musk* dataset [AN07b] has 168 features and 6598 samples, of which 1017 were musks and 5581 non-musks. For this test we found that the best configuration for RRD used K-means discretization, the Cramer's ϕ_c statistic, and random forests as the classifier. Table 2.9 shows a comparison between RRD and the best published results. RRD achieved the best result with $97.33\% \pm 0.34\%$ accuracy, using 37.45 ± 2.35 features.

Table 2.9: Musk dataset: comparison table. RRD is the best RRD configuration using K-mean discretization, Cramer's ϕ_c statistic and random forest classifier

N. feat. sel.	Accuracy	Algorithm	Ref.
37	**97.33%**	**RRD**	**[*]**
65	97.06%	PFA + 1NN	[WYT+07]
65	97.06%	PFA + SVM	[WYT+07]
25	96.35%	FCBF-P	[YL03a]
6	92.4%	HSMB	[LO10]
2	88.2%	FCBF	[YL04a]
4	88%	GS	[MT00]
Na	87.4%	mi - SVM	[ATH03]

The microarray dataset used is the colon cancer dataset [ABN+99] which consists of 2000 features with 22 normal and 40 cancerous tissues. This is high-dimensional continuous data with few samples, making it very different to the other datasets we used. Expert opinion is available on which genes should be selected, so as a first experiment we applied RRD to the entire dataset to select features. There were 2 combinations of RRD that were of interest, one giving a small but well referenced set of genes. And the other beating published results, including recent ones. These are initial results and hopefully with further investigation we will be able to establish one robust configuration for RRD. **RRDa** with k-mean discretization and Tschuprow's T statistic selected 6 genes, 5 of which were in the top 40 identified by Chen et al. [CN09], 4 of which were in the top 10, 5 of which were in the top 20 noted by Han et al. [HLC+11], 3 in the top 10. The genes selected by RRD were:

- *M63391 gene 1* "*Human desmin gene, complete cds*". This gene was selected by both Han et al. and Chen et al. [CN09, HLC+11].
- *Z50753 gene 1* "*H.sapiens mRNA for GCAP-II/uroguanylin precursor*". This gene was selected by Li et al., Shevade et al., Han et al. and Chen et al. [CN09, HLC+11, LWH08, SK03].
- *R87126 3' UTR 2a 197371* "*MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)*". This gene was selected by Li et al., Shevade et al., Gonzalez et al., Mahata et al., Han et al. and Chen et al. [CN09, NM09a, HLC+11, LWH08, MM07, SK03].
- *T86473 3' UTR 1 114645* "*NUCLEOSIDE DIPHOSPHATE KINASE A (HUMAN)*". This gene was selected by Han et al. and Chen et al. [CN09, HLC+11].

- *U22055 gene 1 "Human 100 kDa coactivator mRNA, complete cds*". This gene was selected by Chen et al. [CN09].
- *U25138 gene 1 "Human MaxiK potassium channel beta subunit mRNA, complete cds*". This gene was selected by Han et al. [HLC$^+$11].

RRDb with equal frequency discretization and Contingency Coefficient C statistic selected 9 genes, 4 of which were in the top 40 identified by Chen et al. [CN09], 3 of which were in the top 20 noted by Han et al. [HLC$^+$11]. The genes selected by RRD were:

- *T74556 3' UTR 1 84680 "ATP SYNTHASE ALPHA CHAIN, MITOCHONDRIAL PRECURSOR* ".
- *Z50753 gene 1 "H.sapiens mRNA for GCAP-II/uroguanylin precursor*". This gene was selected by Li et al., Shevade et al.,Han et al. and Chen et al. [CN09, HLC$^+$11, LWH08, SK03].
- *H40137 3' UTR 2a 191720 "IRON-RESPONSIVE ELEMENT BINDING PROTEIN (Homo sapiens)*."
- *M80815 gene 1 "H.sapiens a-L-fucosidase gene, exon 7 and 8, and complete cds*".
- *U22055 gene 1 "Human 100 kDa coactivator mRNA, complete cds*". This gene was selected by Chen et al. [CN09].
- *T47377 3' UTR 1 71035 "S-100P PROTEIN (HUMAN)*". This gene was selected by Chen et al. [CN09].
- *R39209 3' UTR 2a 23464 " HUMAN IMMUNODEFICIENCY VIRUS TYPE I ENHANCER-BINDING*".
- *M36634 gene 1 "Human vasoactive intestinal peptide (VIP) mRNA, complete cds*". This gene was selected by Mahata et al.,Han et al. and Chen et al. [CN09, HLC$^+$11, MM07].
- *H08393 3' UTR 2a 45395 "COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)*". This gene was selected by Chen et al. [CN09].

Secondly, we used cross-validation to test the accuracy of both versions of RRD on the dataset. Table 2.10 shows a comparison between RRDa, configured with k-mean discretization, Tschuprow's T statistic and Naive Bayes classifier, and RRDb configured with equal frequency discretization, Contingency Coefficient

C statistic and Naive Bayes classifier, and published results. On the colon cancer dataset RRDa achieved $87.50\% \pm 5.94\%$ accuracy, with 8.85 ± 0.99 features, which was the second best result, and RRDb achieved $84.21\% \pm 7.82\%$ accuracy, with 5.79 ± 0.54 features, which was the fourth best result.

Table 2.10: Colon cancer dataset: comparison table.

N. feat. sel.	Accuracy	Algorithm	Ref.
15	87.90%	MPE + SVM	[MM07]
9	**87.50%**	**RRDb**	**[*]**
2000	87.10%	NPS+LogitBoost	[DB03]
18	86.40%	BKPR	[Cha09]
33	85.48%	ReliefF + SVM	[KŁ16]
10	85.48%	mRMR + SVM	[KŁ16]
6	84.21%	RRDa	*
13	81.10%	SVM + RFE	[Cha09]

2.8 Conclusion

We proposed a new filter-based feature selection algorithm with several advantages over existing methods: it can use a variety of statistics; it can handle combinations of nominal, ordinal and numerical data; it takes both relevance and redundancy into account; and it automatically decides how many features to select without the need for a threshold or cut-off. In experiments on credit data it outperformed published methods, especially using a new proposed statistic.

We show that by investigating the relationship between features, and between feature and target, we can eliminate dominated features. This helps to reduce both overfitting and model complexity, thus improving the classifiers' prediction power. Our algorithm, *RRD*, works well as it retains the most important features, and from the experimental work, we learn that our dominance idea has great potential as a feature selection method. Although most of our results are on credit scoring data, where any improvements in classification accuracy is of great value to the financial sector, *RRD* can be used as part of the pre-processing pipeline for most classification problems. By using *RRD* as the feature selection method, the parameters in the classification problem can be reduced, and the model achieving better generalization.

After showing promising results on high dimensional data, *musk* and *colon* datasets, we further investigated on other microarray data. The results showed that as the datasets increased in dimensionality, *RRD* was inclined to retain

more features than required, which caused the classifiers to lose accuracy due to over-fitting. It was also found to increase significantly in computational cost as the dataset dimensionality grew.

In the next chapter, we propose an improved version of the algorithm. It increases the efficiency of subsetting the relevant features in the data, where the first part of the algorithm rapidly removes dominated features, retaining only a small subset of important ones. Then, *RRD* filters through the kept features to select the optimal un-dominated features. We focus, in the next chapter, purely on microarray datasets since their dimensionality ranges from $2000 - 24188$, which we believe shows the robustness of this novel filter feature selection method.

Chapter 3

Fast Relevance-Redundancy Dominance: Feature Selection for High Dimensional Data

3.1 Motivation

This chapter continues the work on filter feature selection, extending the work on using dominance to select a subset of features for model construction, but in this chapter we will focus on microarray data as the classification problem to detect various types of cancer. This reduces dimensionality which is important for simplification, efficiency and reducing over-fitting. Filter-based methods are the most scalable, rating features by their relevance to the target variable via appropriate statistics. In chapter 2, we proposed a filter feature selection method, called Relevance-Redundancy Dominance (RRD) with useful properties (no threshold setting, adaptability to any statistics etc.), but with a poor scalability. In this chapter, we present a scalable version of RRD, called Fast Relevance-Redundancy Dominance which has the same properties as RRD while improving scalability. To show the effectiveness of the proposed approach we have carried out extensive numerical experiments on high dimensional datasets (DNA microarray datasets) which shows that it outperforms state-of-the-art algorithms.

3.2 Introduction

The task of feature selection is to select a subset of the original features present in a given dataset that provides most of the useful information. Hence, after selection has taken place, the dataset should still have most of the important information still present. As mentioned in the previous chapter, Chapter 2, the methods are usually categorized as filter, wrapper or embedded [GGNZ08]. While filter methods rely on the general characteristics of the training data to select features, independently of any predictor, wrapper methods involve optimizing a predictor as part of the selection process; and embedded methods try to combine advantages of both. The advantages of filter methods over wrapper and embedded methods are that they are fast, independent of the classifier/predictor method, scalable and easy to interpret. For this reason, they are particularly suitable for large datasets.

Due to these advantages, many filter methods have been proposed in the past and recent years. The RELIEF algorithm [KR92] estimates the quality of attributes according to how well their values distinguish between instances that are similar to each other, but was initially limited to two-class problems. An extension, ReliefF [Kon94], not only deals with multi-class problems but is also more robust and capable of dealing with incomplete and noisy data. The main drawback of the Relief family methods is that they select features based only on relevance and do not remove redundant features.

Correlation-based Feature Selection (CFS) [Hal99b] is a simple filter algorithm that ranks feature subsets according to a correlation-based heuristic evaluation function. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. However, redundant features are screened out as they are highly correlated with one or more of the remaining features. Moreover, there exists an improved CFS version called Fast Correlated-Based Filter (FCBF) method [YL03b] based on symmetrical uncertainty (SU) [PTVF88]. This version is designed for high-dimensionality data and has been shown to be effective in removing both irrelevant and redundant features, but on the other hand it fails to take into consideration interactions between features.

The INTERACT algorithm [ZL09] uses the same goodness measure as the FCBF filter [PTVF88] but also includes the consistency contribution (c-contribution). The c-contribution of a feature indicates how significantly the elimination of

that feature would affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. If the c-contribution of a feature is less than a given threshold the feature is removed, otherwise it is selected.

Finally, Minimum Redundancy-Maximum Relevance (MRMR) [PLD05b] is a heuristic framework which minimizes redundancy, using a series of measures of relevance and redundancy to select promising features for both continuous and discrete data sets.

In this chapter, we extend the work of RRD from the previous chapter, Chapter 2. Although RRD has many advantages compared to state-of-the-art methods: (i) it can be applied to mixed data types, (ii) it can be used with a wide variety of statistics and (iii) it requires no threshold for choosing a feature subset. However, it shows poor scalability on high-dimensional data. In order to overcome this drawback, in this chapter, we propose a fast version of RRD called Fast Relevance-Redundancy Dominance (FRRD) feature selection, with a simple feature elimination strategy also based on relevance and redundancy. The key differences between RRD and FRRD are (i) FRRD rapidly reduces the number of features in the dataset, based on group dominancy, whereas RRD removes features singly, (ii) FRRD requires a predefined reduction rate and a set number of retained features for step 1 of the algorithm, but in contrast, RRD has no user set parameters. These 2 main differences help to make FRRD very suitable for feature selection on high dimensional data. The results of the proposed method, achieved on extensive numerical experiments using microarray data, are compared and contrasted with other methods in literature.

3.3 Related Work

Feature selection is a prevalent part of pre-processing the inputs for a machine learning model, and its importance is well documented. The benefits of selecting a subset of informative features include faster model learning, better generalization, reduced computational costs and with less variables, hopefully, better model explainability.

[BCSMAB12] proposed an ensemble of filters and classifiers approach. They used 5 filter methods, individually, to extract the important features in the dataset, used an ensemble classifier – C4.5 decision tree, naive Bayes and

instance-based learner IB1 – on each subset to get a prediction and then using majority voting to get a final prediction. They conclude that their ensemble approach is more robust and stable compare with using a single filter-classifier method. [BCSMAB11] followed a very similar idea but, instead of classifying each subset of features from the filter methods, they combined the features into a single subset and using the ensemble classifier predicted the output. This approach didn't yield as good result as the first method, even with reduction thresholds they enforced on the union of the filter feature subsets.

[NM09b] introduced an entropic filtering algorithm (EFA) that finds a subset of features that jointly maximizes the normalized multivariate conditional entropy in respect of the classification ability. The authors use class-attribute interdependence maximization to discretize the datasets, similar to our preprocessing, where we experimented using various "binning" methods such as "equal-width". [BM10] used an information theory based filter approach called min-Interaction Max-Relevance (mIMR). Their method was based on interaction (link between causal discovery and feature selection) which has shown to be informative about how relevant a selected subset is, as well as its causal correlation with the target. [MM13] method was a variation of the well-documented minimum redundancy maximum relevance (mRMR). The authors used mutual information both between a feature and the target as the measurement of relevance, and between features as the redundancy measurement. A drawback of this method was that the user first had to state the number of informative features the algorithm retained, which could be difficult without domain knowledge.

[LJL+14] used a spectral graph theory method called locality sensitive Laplacian score, which takes into account discriminative information into local geometrical architecture, similar to a *k-means* method. This means that the algorithm, in a supervised manner, simultaneously minimizes the local within-class information and maximizes the local between-class information. [MS11] proposed 2 feature selection methods; the first used *k-means* to cluster the features before ranking them according to their signal-to-noise ratio (SNR), and the second ranked the unclustered features with respect to their SNR. Both methods used a user defined number of top-ranked informative features as the input to the classifier. The authors results show that the *k-means* method achieved better classification accuracy, which can be seen as a type of discretization preprocessing.

3.4 Classifiers and Datasets

3.4.1 Microarray Datasets

Table 3.1: Microarray datasets: name of dataset, N is the number of features, n the number of samples, percentage of minimum $\%$min and majority $\%$maj classes, the imbalance ratio IR and the maximum Fisher's discriminant ratio F1.

Dataset	N	n	(%min,%maj)	IR	F1
Alizadeh (DLBCL)	4026	47	(48.94,51.06)	1.04	120.35
Alon (Colon)	2000	62	(35.48,64.52)	1.82	22.45
Beer (Lung)	7129	96	(10.42,89.58)	8.6	484.02
Freije (Gli85 Brain)	22283	85	(30.59,69.41)	2.27	90.37
Gordon (Lung)	12533	181	(17.13,82.87)	0.21	187.37
Golub (Leukaemia)	7129	72	(34.72,65.28)	1.88	63.74
Petricoin (Ovarian)	15154	253	(35.97,64.03)	1.78	333.63
Pomeroy (CNS)	7129	60	(35,65)	1.86	16.92
Shipp (Lymph)	7129	77	(24.68,75.32)	3.05	103.32
Singh (Prostate)	12600	136	(43.38,56.62)	1.31	36.82
Spira (Lung)	19993	187	(48.13,51.87)	1.08	19.28
Veer (Breast)	24188	95	(46.32,53.68)	1.16	29.93
Pre-established training & test samples					
Singh (Prostate) Train	12600	102	(49.02,50.98)	1.04	66.46
Singh (Prostate) Test	12600	34	(26.47,73.53)	2.78	687.23
Veer (Breast) Train	24188	78	(43.59,56.41)	1.29	25
Veer (Breast) Test	24188	19	(36.84,63.16)	1.71	148.17

FRRD was evaluated on 12 microarray datasets, which are available on the Kent Ridge Biomedical Data Set Repository [LL02] and Arizona State University [Ari]. Each dataset has a binary target variable. Table 3.1 provides an overview of the 12 datasets. The datasets chosen have a variety of characteristics: they are high dimensional, from 2000 up to 24188 features; from quite well balanced (with 1:1 being the ideal value) to fairly unbalanced (as shown by IR ratio); have a wide variety of complexity (as shown by F1 ratio), and finally like all microarray datasets, they are very challenging as all of them have very few samples (from 47 up to a maximum of 253). Furthermore, two of these datasets are pre-established training and testing samples which are detailed at the bottom of Table 3.1. The tables show the author's name (who originally used the dataset), along with the type of data being examined, the number of features (genes), and the number of samples. It also shows the percentages of the minimum and majority classes, as well as the imbalance between the classes. It can be seen that most of the datasets have an imbalance of between 1 and 2, which is reasonable. However, there are a few outside this range; the

most notable are Beer and Gordon, both having one class more than 8 times larger than the other. The F1 column shows the maximum Fisher's discriminant ratio of each dataset [HB02]. This shows the complexity of the datasets, which measures the overlapping within the dataset. The lower the F1 score, the more overlapping within the dataset, hence making it a greater challenge for classification. The two datasets with the lowest F1 values are Pomeroy and Spira. It will be shown in the results section, that these 2 datasets are harder to correctly classify, thus having poorer classification results when compared to the other datasets. Feature selection or Gene selection has been the subject of several papers ([BCSMAB11], [MM13], [NM09b]), making it an ideal testbed for our method. Microarray data has a distinctive characteristic of high dimensionality (in our case 24188 genes) with a small sample size, which can be a very challenging problem for classification methods. To compare our algorithm with others in the literature, we have run FRRD on each dataset, then we have compared our results with the results provided by other authors using the same model validation techniques and classifiers used in those papers. We did not re-implement other authors' techniques, but instead took results from their published works.

3.4.2 Classifiers

We used 5 classifiers: Random Forests, Naive Bayes, Support Vector Machine (SVM), C4.5 Decision Tree and k-nearest neighbours (KNN). In the KNN algorithm, we use 1, 3, and 5 as the k value to compare to related work results. We chose these five methods because they are widely used in the microarray classification research area. The Random Forests and Naive Bayes classifiers have been detailed in Section 2.4.4, while the remaining classifiers are explained below:

- ***Support Vector Machines*** (SVM) are widely used for binary classification problems, but can also be used in multi-class task using an *one-vs-all* approach. SVM try to find the most optimal hyper-plane, which separates the classes in the data (see Figure 3.1). The weights of the SVM are adjusted to find those that maximize the distances (margin) between the classes and separating line. The greater the margin, the more confident the prediction of the SVM. To create an optimization problem, instead of looking for a maximization (which can not be solved using gradient descent due to being non-convex), by taking the inverse and square of it we

get a minimization problem. This can be solved using Lagrange Optimization [JL12].

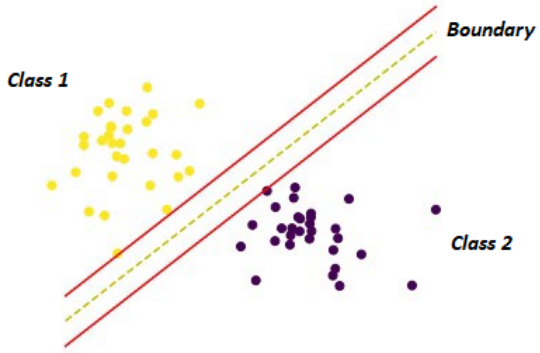

Figure 3.1: This figure shows how a SVM finds an optimal separating line, with large enough margins to separate the data into its classes.

- ***C4.5 Decision Tree*** classifiers uses information gain and entropy to split the data into smaller and smaller subsets, while at the same time creating a decision tree based on the splits (see Figure 3.2). A decision tree looks like a flowchart, where the features are split using a set of if-else decision rules. When the tree is fully developed, the best feature to split the data was the root node, followed by decision nodes where the data is further split along these branches. Finally, a leaf node is the node at which classification is performed. A pitfall of decision trees is that they can learn the training data too well, and thus over-fit on it. To help prevent this, the size of the tree is reduced by turning some branches into leaf nodes, and removing the leaf nodes under the original branch, through pruning.
- k***-nearest neighbours*** (KNN) is a non-parametric (makes no assumptions) lazy-learner (makes no generalizations) algorithm. To classify unseen data points, KNN looks at k closest points to the new point and classifies it in a majority voting way (see Figure 3.3). For this work we look at 1, 3 and 5 for the values of k. It is usual to select an odd value for k to prevent a draw occurring. The standard distance measurement used in KNN is the Euclidean distance between the test point and the k points in the training data.

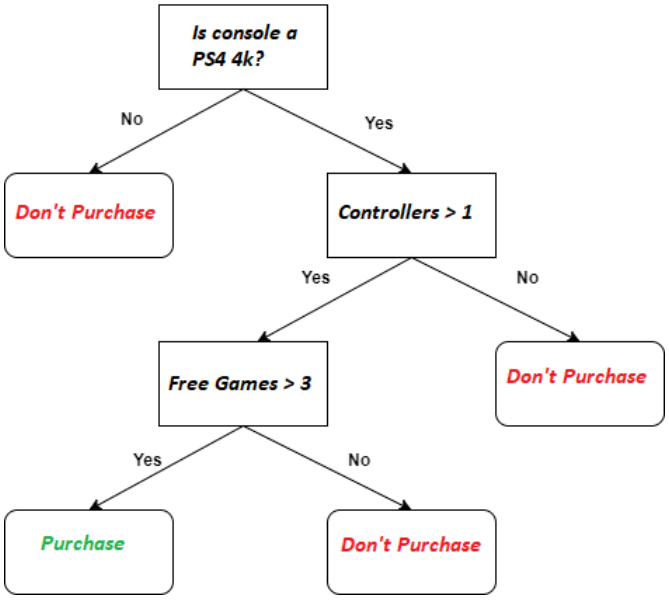

Figure 3.2: This figure shows how a C4.5 Decision Tree splits the data into smaller subsets (branches), until reaching a decision node for the prediction.

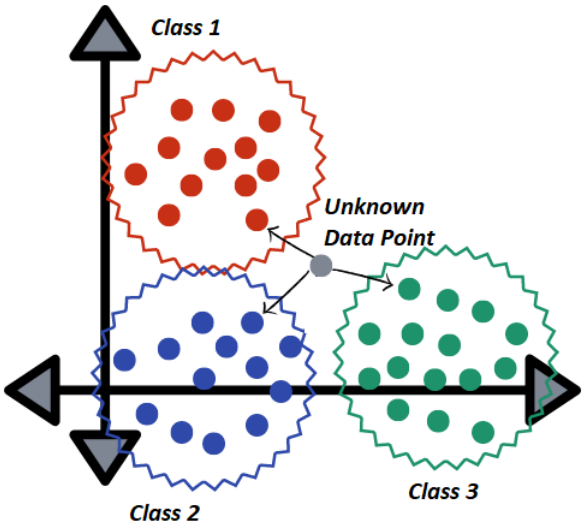

Figure 3.3: This figure shows how a KNN classifier, with $k = 3$, selects the class of an unknown test sample using the closest samples in the training data.

3.5 Materials and Methods

3.5.1 Experimental Design

The FRRD method has been evaluated from the perspective of classification performance, comparing it with state-of-the-art algorithms. Papers in the literature, using filter feature selection methods such as ReliefF, INTERACT, Informa-

tion Gain, CFS and Consistency-based filter, use different cross-validation techniques in their analysis, i.e.: 5-fold, 10-fold and leave-one-out cross-validation, as well as different well-known classifiers, namely 3KNN, 5KNN, C4.5, Naive Bayes, IB1, SVM and Random Forest. In order to evaluate the effectiveness of the proposed method, this work uses the same cross-validation and classifier techniques as the other methods in each comparison. All features were discretized using equal-width interval, see Section 2.4.2.

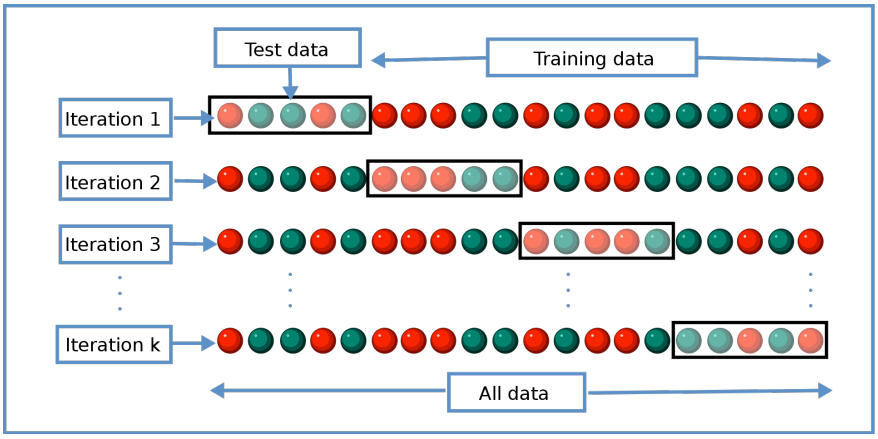

Figure 3.4: An example of a k Cross-Validation splits on the data set.

Figure 3.4 shows a k-fold split on a 2 class dataset, where k in this set of experiments could be 5, 10 or leave one out cross validation (LOOCV). In LOOCV, every sample, individually is selected as the test set, while the remaining samples are the training set, and the number of samples in the data becomes the k in the k-fold. The dataset is split into training and testing sets in a stratified approach to account for any imbalances in the dataset.

3.5.2 Fast Relevance-Redundancy Dominance

Fast Relevance-Redundancy Dominance (FRRD) is an extension of the original *Relevance-Redundancy Dominance* (RRD) [BMP16], from the previous chapter (Chapter 2), and although it uses the same principal idea of dominance, unlike RRD it is highly scalable. It should also be noted that RRD is step 2 in the FRRD algorithm, but, due to the significantly reduced number of features as the input, it is extremely fast to find a good subset of important features. RRD was tested on high dimensional data, in particular the colon [ABN+99] and leukaemia [LL02] micro array datasets, where experiments showed RRD retaining a lot more features than needed. For the colon dataset, RRD kept between $40 - 50$ features, while, for the leukaemia dataset, between $80 - 100$ features were

retained. As a result of this, the computational time increased, along with the classification accuracy decreasing due to over-fitting. FRRD was proposed, based on the same idea as RRD, to overcome the pitfalls of RRD.

Fast Relevance-Redundancy Dominance is a univariate filter-based feature selection method, which can use any suitable statistic to select a good (relevant and non-redundant) subset of features from a dataset. Before detailing the proposed approach, we first remind the reader of some definitions previously defined in Section 2.5.2. Let f and f' denote features, t the target variable, and s the chosen statistic (which may be symmetric or asymmetric), then:

Definition 3.5.1. *f **dominates** f' if $s(f,t) > s(f',t)$ and $s(f,f') > s(f',t)$.*

Definition 3.5.2. *f' is **redundant** if it is dominated by some f that is not dominated by any other feature.*

Algorithm 2 Step 1 of FRRD - section of the algorithm that rapidly reduces the number of features in the dataset

1: given features F and target t
2: $N \leftarrow |F|$
3: $K \leftarrow \log_2 N$
4: $\alpha \leftarrow 10\%$
5: **for all** $f \in F$ **do**
6: $\quad x_{ft} \leftarrow s(f,t)$
7: **end for**
8: $F \leftarrow ([\alpha N] + 1)K$ of the top most relevant features from the set F
9: $S \leftarrow \emptyset$
10: **while** $|S| < K$ **do**
11: $\quad \hat{f} \leftarrow \arg\max_{f \in F} x_{ft}$
12: $\quad R \leftarrow \{(\alpha|F|)$ greatest features $f \mid f \in F \wedge s(\hat{f},f) > x_{ft}\}$
13: $\quad F \leftarrow F \setminus (R \cup \{\hat{f}\})$
14: $\quad S \leftarrow S \cup \{\hat{f}\}$
15: **end while**
16: **return** S

The following bullet points give a line-by-line description of Algorithm 2:

- In line 1 we define the features F and the target t from the dataset.
- Line 2 we initialize N to the number of features in the set F.
- Line 3 we initialize K to the log of N to the base 2, which is the number of features we want to retain from step 1 of the algorithm.

- Line 4 we initialize α to 10%, which is the rate at which we remove dominated features.
- Lines $5-7$ are a loop; this is exited (line 7) when the all the features of the dataset have been analyzed.
- The purpose of this loop is to calculate a given statistic between each feature and the target.
- In line 8 we reduce the set of features F, retaining only the number of most relevant features we need to complete the feature selection.
- Line 9 is where we initialize S as an empty set, which we will fill with the selected features.
- Lines $10-15$ are a loop; this is exited (line 15) when the set of features S no longer contains less than K features.
- This loop selects the important features, and removes groups of the dominated features.
- Line 11 selects the feature ($\hat{f}$) which has the greatest relevance, with respect to the target.
- Line 12 creates a set of features R, which are dominated by the current most relevant feature ($\hat{f}$), and are removed in groups of features.
- Line 13 is where the set of dominated features R, and the current most relevant feature ($\hat{f}$) are removed from the set of features F.
- In line 14 we insert the current most relevant feature ($\hat{f}$) into the pre-initialized set S.
- Line 16 ends the algorithm and returns the selected set of feature S.

The following bullet points give a line-by-line description of Algorithm 3:

- In line 1 we define the remaining features S and the target t from the dataset.
- Lines $2-4$ are a loop; this is exited (line 4) when all the features of the set S have been analyzed.
- The purpose of this loop is to calculate a given statistic between each of the remaining features and the target.

Algorithm 3 Step 2 of FRRD - this section of the algorithm is RRD (see Section 2.5.2), but on the reduced subset S

1: given remaining features S and target t
2: **for all** $f \in S$ **do**
3: $x_{ft} \leftarrow s(f, t)$
4: **end for**
5: $T \leftarrow \emptyset$
6: **while** $S \neq \emptyset$ **do**
7: $\hat{f} \leftarrow \arg\max_{f \in S} x_{ft}$
8: $R \leftarrow \{f \mid f \in S \wedge s(\hat{f}, f) > x_{ft}\}$
9: $S \leftarrow S \setminus (R \cup \{\hat{f}\})$
10: $T \leftarrow T \cup \{\hat{f}\}$
11: **end while**
12: **return** T

- Line 5 is where we initialize T as an empty set, which we will fill with the selected features.
- Lines $6 - 11$ are a loop; this is exited (line 11) when the set of features S contains no more features f.
- This loop selects the important features, and removes the dominated features.
- Line 7 selects the feature ($\hat{f}$) which has the greatest relevance, with respect to the target.
- Line 8 creates a set of features R, which are dominated by the current most relevant feature ($\hat{f}$).
- Line 9 is where the set of dominated features R, and the current most relevant feature ($\hat{f}$) are removed from the set of features S.
- In line 10 we insert the current most relevant feature ($\hat{f}$) into the pre-initialized set T.
- Line 12 ends the algorithm and returns the selected set of feature T.

FRRD has 2 parameters, α, a reduction-rate and K, the number of features desired (K is user defined, and in this thesis was set to $\log_2 N$, N being the total number of features). FRRD is a two-step algorithm, the first comprises of the new scalable approach of the algorithm, while the second is the original RRD (see Section 2.5.2) but on the reduced set of features selected by Algorithm 2. Step one of FRRD is expected to select all non-redundant features using the

algorithm shown in Algorithm 2. Then, we calculate the number of features F which FRRD needs to retain to complete the feature selection. This is $\alpha N + 1$ which is then multiplied by the number of features desired. We initialize an empty set S. After this, in the while loop, we select the feature $\hat{f} \in F$ with greatest relevance , generate the set R of $f \in F$ that are the α most dominated by $\hat{f}$, remove $R \cup \{\hat{f}\}$ from F, and add $\hat{f}$ to S. The last four steps in the loop are repeated until the number of features desired K, is reached, then we return the set S of selected features.

First we precompute the statistics $x_{ft} = s(f, t)$ between each feature f and the target variable t. Once step 1 (Algorithm 2) has selected the *K* features, named S in step 2, FRRD runs in order to select all non-redundant features using the procedure shown in Algorithm 3. We use the computed statistics of $x_{ft} = s(f, t)$ between each of the remaining features f and the target variable t. We initialise the set of selected features T to the empty set $\emptyset$. Then we select the feature $\hat{f} \in S$ with greatest relevance $x_{\hat{f}t}$, generate the set R such that $f \in S$ that are made redundant by $\hat{f}$, remove $R \cup \{\hat{f}\}$ from S, and add $\hat{f}$ to T. The last four steps in the loop are repeated until S is empty, then we return the set T of selected features. In theory at the end of step 2 the value K does not have to equal the size of the set T, as it is possible that a feature in S may have been dominated by another feature in that set.

Note that FRRD typically does not compute all s-values between features (for example a full correlation matrix), nor does it sort a complete list of features. This is because after a feature has been removed from F no further computations on it need be performed. This means, the worst case time complexity is $O(\alpha N K^2 M)$ for N, total number of features, K desired number of features and M samples. It is possible to construct datasets for which Algorithm 3 removes no features at all, or removes all but one, but in practice we find that it usually generates a small subset of features. Furthermore, FRRD is highly flexible in terms of the statistics it can use. In fact, the statistic can be symmetric $s(f, f') > s(f', f)$ (for example correlation or mutual information) or asymmetric (for example Goodman's λ [GK79]), once the correct order of f and f' is maintained.

Finally, it should be pointed out that FRRD assumes that we can compute statistic $s(x, y')$ for all $x, y \in F \cup \{\hat{t}\}$. Thus before we can apply FRRD, if the target and/or features have mixed types (numerical, ordinal, nominal) they must first be pre-processed so that they are all of the same type. Discretization is used

to transform continuous data to discrete form, which involves putting the continuous data put into 'bins', using well-known methods such as equal-width. This reduces and simplifies the data, improving the computational efficiency, and it has also been shown to increase the accuracy of filter algorithms [BCSMAB10]. Another advantage, showing the versatility of FRRD, is its ability to handle mixed data types (transformed to discrete variables) by using any suitable statistic whether they are correlation criteria (such as Pearson's correlation coefficient) or information-based (such as mutual information) depending on the type of data being analysed.

3.6 Stability and Scalability

One of the challenges of feature selection is to select the smallest set of features which still classifies test data accurately. The stability of the filter algorithm is its ability to repeatedly select the same or a similar subset of features in each k-fold. A variation of the formula used to calculate the Tanimoto distance between 2 sets (shown in Equation 3.1), introduced by Kalousis [KPH07], is used in this thesis, and the results are compared to Bolon et al in Table 3.2 [BCSMAB12]. A greater Tanimoto distance indicates a better stability measure in the method. The results show that FRRD is more stable than the newly proposed method by Bolon et al, scoring higher than it in all eight datasets. FRRD and Information Gain (IG) both achieved the best stability result in three datasets each, while ReliefF scored best on the other 2 datasets. We define the Tanimoto distance as;

$$T(S_i\,, S_j) = 1 - \frac{|S_i| + |S_j| - 2|S_i \cap S_j|}{|S_i| + |S_j| - |S_i \cap S_j|} \tag{3.1}$$

where:
S_i and S_j are two subsets selected from different k folds of the cross-validation

Table 3.3 shows FRRD's computational time, which is compared to that of other well-known filter methods [KŁ16]. The experimental setup was on a single thread core, and it can be seen that FRRD requires much less computational time than the others. It should be noted that RRD's computational time for the colon dataset was slightly over an hour, showing it not to be a very scalable method. In the next section we will show that the classification accuracy of FRRD is also superior to well-known methods.

Table 3.2: Comparison of FRRD's stability measure with the paper [BCSMAB12] using 10-fold cross- validation.

Dataset	FRRD	Ensemble	CFS	Cons	INT	IG	ReliefF
Alon (Colon)	0.512	0.511	0.319	0.138	0.264	0.529	0.675
Golub (Leukaemia)	0.59	0.396	0.213	0.17	0.246	0.654	0.396
Pomeroy (CNS)	0.424	0.266	0.208	0.151	0.182	0.252	0.307
Alizadeh (DLBCL)	0.576	0.38	0.274	0.47	0.232	0.488	0.621
Singh (Prostate)	0.713	0.363	0.34	0.077	0.207	0.322	0.395
Gordon (Lung)	0.704	0.287	0.247	0.126	0.221	0.721	0.605
Petricoin (Ovarian)	0.486	0.476	0.386	0.486	0.262	0.875	0.688
Veer (Breast)	0.358	0.229	0.194	0.061	0.178	0.212	0.301

Table 3.3: Comparison of FRRD time, using single thread core CPU, with the [KŁ16], using leave-one-out cross-validation (in hours:minutes:seconds).

Dataset	FRRD	ReliefF	MRMR	SVM-RFE
Alon (Colon)	0:04:08	0:51:00	0:50:05	1:49:55
Golub (Leukaemia)	0:24:00	1:50:12	1:35:32	13:03:27
Pomeroy (CNS)	0:13:00	57:01:17	0:49:37	8:20:36
Singh (Prostate)	1:39:44	19:47:54	15:22:52	146:23:27
Gordon (Lung)	2:24:48	57:01:17	12:33:52	231:28:22
Veer (Breast)	2:36:49	18:45:00	10:31:17	250:50:00

3.7 Results

FRRD results are compared with a number of different published methods, showing the robustness of the algorithm. We first run FRRD using 5-fold cross-validation to compare with the results of Bolon [BCSMAB$^+$14b] (see Table 3.5). Out of the 18 experiments, FRRD was only beaten once, and that was by 0.09%, on the Ovarian dataset using the C4.5 classifier. On the other experiments, FRRD outperforms the seven well-known filter methods, including mRMR, CFS and FCBF. It should also be noted that FRRD also selected far fewer features than the other methods. Bolon conducted an analysis on pre-established training and testing samples, which, was also performed in this study using FRRD. The results show that using the same split in the datasets, FRRD still achieves better classification accuracy than the well-known methods. Mandal (see Table 3.4) introduced an improved version of mRMR, along with the results of 2 other versions of mRMR, which are compared with FRRD using 10-fold cross-validation [MM13]. On the two microarray datasets selected by Mandal, FRRD achieves 100% accuracy, equalling the improved mRMR on one dataset and being superior to the rest of the experiments on the paper.

[BM10] proposed a method called min-Interaction Max-Relevancy (mIMR) and

Table 3.4: Comparison of FRRD with the [MM13], using 10 fold cross-validation (Accuracy in percentage with number of features selected in brackets).

Dataset	FRRD	Improved mRMR	mRMR (MID)	mRMR (MIQ)
Golub (Leukaemia)	100(8.7)	100(100)	97.22(100)	98.61(100)
Petricoin (Ovarian)	100(13)	99.8(100)	98.2(100)	98.6(100)

Table 3.5: Comparison of FRRD with the paper [BCSMAB$^+$14b], using 5 fold and holdout cross-validation with C4.5, Naïve Bayes and SVM classifiers (Accuracy in percentage with number of features selected in brackets).

Dataset	FRRD	No FS	CFS	FCBF	INT	IG	ReliefF	SVM-RFE	mRMR
C4.5 Classifier									
Alizadeh (DLBCL)	98(12)	70	75(61)	73(35)	70(45)	75(10)	85(10)	82(50)	75(10)
Alon (Colon)	87(11)	74	79(24)	79(14)	79(14)	84(50)	82(50)	80(50)	82(10)
Freije (Gli85 Brain)	91(14)	75	79(141)	82(116)	78(117)	85(10)	85(10)	82(10)	75(50)
Petricoin (Ovarian)	99(8.4)	97	98(33)	99(26)	98(31)	96(10)	99(50)	98(10)	98(10)
Pomeroy (CNS)	78 (13)	58	62(44)	48(33)	55(33)	63(50)	53(50)	65(10)	58(50)
Spira (Lung)	74(14)	65	64(103)	61(51)	59(51)	65(50)	65(10)	65(50)	68(10)
Naïve Bayes Classifier									
Alizadeh (DLBCL)	100(12)	92	90(61)	90(35)	90(45)	94(10)	94(10)	92(10)	94(50)
Alon (Colon)	89(11)	55	85(24)	80(14)	77(14)	79(10)	84(50)	76(50)	80(10)
Freije (Gli85 Brain)	93(14)	84	82(141)	85(116)	82(117)	85(10)	89(50)	88(50)	85(10)
Petricoin (Ovarian)	100(8.4)	93	100(33)	99(26)	100(31)	98(50)	98(50)	99(10)	99(10)
Pomeroy (CNS)	80(13)	60	67(44)	70(33)	70(33)	63(10)	67(50)	70(50)	63(10)
Spira (Lung)	75(1)	63	65(103)	69(51)	64(51)	66(50)	67(10)	71(10)	67(10)
SVM Classifier									
Alizadeh (DLBCL)	100((12)	96	88(61)	81(35)	88(45)	94(10)	94(10)	88(50)	96(50)
Alon (Colon)	90(11)	77	81(24)	84(14)	81(14)	85(50)	85(50)	73(10)	84(50)
Freije (Gli85 Brain)	94(14)	92	88(141)	87(116)	88(117)	91(10)	89(50)	89(50)	89(10)
Petricoin (Ovarian)	100(8.4)	100	100(33)	100(26)	100(31)	100(50)	100(50)	100(10)	100(10)
Pomeroy (CNS)	80(13)	67	62(44)	65(33)	62(33)	67(50)	73(50)	73(10)	70(50)
Spira (Lung)	75(14)	72	64(103)	71(51)	66(51)	72(50)	69(10)	72(50)	68(10)
Pre-established training & test samples									
C4.5 Classifier									
Veer (Breast)	84(15)	74	68(130)	58(99)	79(102)	53(50)	58(10)	58(10)	74(50)
Singh (Prostate)	76(6)	26	26(89)	26(77)	26(73)	29(50)	29(50)	32(10)	41(10)
Naïve Bayes Classifier									
Veer (Breast)	89(15)	37	37(130)	37(99)	37(102)	37(10)	89(50)	68(10)	37(10)
Singh (Prostate)	85(6)	26	26(89)	26(77)	26(73)	26(10)	21(10)	26(50)	26(10)
SVM Classifier									
Veer (Breast)	89(15)	58	68(130)	58(99)	74(102)	79(50)	84(10)	68(10)	68(50)
Singh (Prostate)	100(6)	53	97(89)	97(77)	71(73)	97(10)	97(50)	79(10)	91 (50)

have used 10-fold cross-validation (Table 3.6), and averaged the results of 5 classifiers 1KNN, 3KNN, 5KNN, SVM-L and Random Forest, reasoning that the use of multiple classifiers would reduce the bias of the feature assessment based on a specific classification strategy. Also in this case is possible to see that our method outperforms mIMR along with the five state-of-the-art filters, used in the paper's experimental design.

In [KŁ16] (Table 3.7) the authors analysed 6 microarray datasets using leave-one-out cross-validation, with SVM as the classifier. FRRD was best in all 6

Table 3.6: Comparison of FRRD with the paper [BM10], using 10 fold cross-validation and averages 5 classifier results, 1KNN, 3KNN, 5KNN, SVM-L and Random Forest (Accuracy in percentage).

Dataset	FRRD	mIMR	mRMR	DISR	CMIM	MB	RANK
Alon (Colon)	88.71	81.66	80.02	80.6	78.41	77	79.45
Golub (Leukaemia)	99.44	94.27	94.23	93.85	93.52	89.24	94.13
Alizadeh (DLBCL)	99.15	93.35	93.99	94.3	92.79	76	93.87
Beer (Lung)	100	96.25	96.3	97.12	92.6	98.7	97.25
Pomeroy (CNS)	80.33	50.88	50.18	50.05	52.01	44.22	49.34
Shipp (Lymph)	98.96	82.37	82.12	83.79	83.07	75.92	82.24

experiments, with far superior accuracy on the CNS dataset. This experiment shows that using another cross-validation technique, FRRD can still perform better than other commonly used filter methods such as mRMR, SVM-RFE and FeliefF.

Table 3.7: Comparison of FRRD with the paper [KŁ16], using leave-one-out cross-validation and SVM classifier (Accuracy in percentage with number of features selected in brackets).

Dataset	FRRD	No FS	ReliefF	MRMR	SVM-RFE
Alon (Colon)	90.32(9.74)	83.87	85.48(33.29)	81.10(12.71)	85.48(9.98)
Golub (Leukaemia)	98.61(13)	98.61	66.67(57.97)	95.83(7.76)	94.44(6.68)
Pomeroy (CNS)	93.75(9.55)	66.67	63.33(18.62)	65.00(49.05)	58.33(14.87)
Singh (Prostate)	93.94(13.99)	91.91	72.06(87.17)	69.12(58.84)	88.24(87.37)
Gordon (Lung)	100(9.29)	99.45	97.24(97.06)	98.34(3.95)	98.90(5.69)
Veer (Breast)	75.79(14)	69.07	64.95(76.69)	70.10(31.65)	67.01(19.90)

The final set of results, again from [BCSMAB12] (Table 3.8), uses 10-fold cross validation with C4.5, Naive Bayes and IB1 classifiers. In this experiment FRRD is superior on all 24 experiments, except for 2, where it was equal best with 100% accuracy. FRRD scored 100% accuracy on 9 out of the 24 experiments, with both Naive Bayes and IB1 classifiers performing best, each of which achieving 100% accuracy on half of the datasets they analysed. In summary, FRRD has been shown to out-perform all other filter methods mentioned in this paper, using a variety of different experimental set-ups, which we believe shows the robustness of FRRD. This is due to FRRD's ability to select the most dominant features, which are also the important features within the data. The stability of FRRD was examined and was quite competitive when compared to other filter techniques. Finally, the scalability of FRRD was explored, showing that FRRD is a very scalable method, with the potential to handle very high dimensional data.

Table 3.8: Comparison of FRRD with the paper [BCSMAB12] using 10 fold cross- validation with C4.5, Naïve Bayes and IB1 classifiers (Accuracy in percentage with number of features selected in brackets).

Dataset	FRRD	Ensemble	No FS	CFS	Cons	INT	IG	ReliefF
C4.5 Classifier								
Alon (Colon)	88.71(11)	86.9	76.19	80.71	83.57	89.05	85.71	77.38
Golub (Leukaemia)	100(13)	88.04	76.96	83.75	85.18	83.75	87.5	88.75
Pomeroy (CNS)	83.33(8.2)	63.33	55	60	58.33	61.67	63.33	58.33
Alizadeh (DLBCL)	97.87(12)	79.5	77.5	73	85.5	76.5	75.5	75.5
Singh (Prostate)	93.38(14)	88.19	80.22	77.03	83.08	81.54	90.6	89.73
Gordon (Lung)	98.9(14)	97.25	93.92	93.92	92.78	93.33	98.89	98.33
Petricoin (Ovarian)	99.21(8.7)	98.8	95.66	97.2	98.43	97.23	98.02	97.65
Veer (Breast)	75.79(15)	63.78	61	61.67	60.22	59.89	66.67	58.89
Naïve Bayes Classifier								
Alon (Colon)	91.94(11)	83.81	52.14	81.90	80	80.71	77.14	85.48
Golub (Leukaemia)	100(13)	95.89	98.75	97.14	89.46	95.89	95.89	95.71
Pomeroy (CNS)	81.67(8.2)	70	63.33	61.67	53.33	63.33	58.33	66.67
Alizadeh (DLBCL)	100(12)	93.5	91.5	96	85.5	94	91.5	95
Singh (Prostate)	94.85(14)	58.13	55.22	63.3	60.99	69.07	57.2	59.73
Gordon (Lung)	100(14)	100	97.78	98.89	91.17	98.89	100	96.67
Petricoin (Ovarian)	100(8.7)	99.2	93.63	99.6	99.22	99.6	97.65	96.05
Veer (Breast)	80(15)	53.44	51.89	51.67	56.78	48.44	56.67	71.33
IB1 Classifier								
Alon (Colon)	88.71(11)	80.95	73.38	80.71	73.57	77.14	79.05	74.29
Golub (Leukaemia)	100(13)	94.46	90.54	94.46	90.54	95.89	93.39	93.04
Pomeroy (CNS)	81.67(8.2)	63.33	48.33	50	58.33	50	53.33	58.33
Alizadeh (DLBCL)	100(12)	96	76	93.5	81.5	91.5	93.5	93.5
Singh (Prostate)	94.12(14)	87.47	79.89	78.96	79.89	79.95	83.13	87.42
Gordon (Lung)	100(14)	98.89	95	99.44	92.78	99.44	98.89	97.81
Petricoin (Ovarian)	100(8.7)	100	94.46	100	99.6	99.2	97.26	98.82
Veer (Breast)	77.89(15)	71.89	60.22	63.78	61.67	67.78	71.89	69

3.8 Conclusion

We proposed a new fast filter-based feature selection algorithm called Fast Relevance-Redundancy Dominance, FRRD, which has similar advantages to RRD (in the last chapter) over existing methods, except that the user needs to decide how many features to retain. In experiments on microarray data, FRRD outperforms the results of published methods, while being competitive with their stability scores, and has been shown to be a very scalable filter technique.

We showed our proposed algorithm, FRRD, works very well by rapidly reducing the number of features in the first part of the algorithm, making it very suitable to high dimensional data, with the second part of the algorithm ensuring none of the retained features are dominated by another feature. FRRD shows us that the dominance idea can be adapted for high dimensional data, in the case of

this chapter microarray data, and maintain classification accuracy by selecting relevant features. The results of this chapter displays how FRRD helps the classifiers to correctly identify cancerous genes which has huge benefits to society and cancer research. It should also be noted that FRRD is not restricted to a particular research area, rather it should be used as part of the pre-processing pipeline, and would work similarly using other high dimensional datasets.

Although being a highly scalable, and accurate, feature selection method, FRRD's main shortcoming is the requirement of a user-defined number of dominant features to retain in step 1 of the algorithm. The threshold-free element of RRD (see Section 2.5.2) from chapter 2 was a nice novel idea, and we propose a final extension of our feature selection methods, *improved Relevance-Redundancy Dominance with Dominant Genetic Algorithm* in the next chapter, Chapter 4. A lot of related work in this area, and in feature selection in general, shows that using a genetic algorithm (GA) for feature selection can help the classifier to have better accuracy, and also be more efficient by selecting less features. We will take the scalability of FRRD, the dominance idea of RRD to select a *small* subset of the data, in an unsupervised manner, which will then be further reduced in a *tournament-style* genetic algorithm.

Chapter 4

IRRD-GA: Gene selection for microarray data using a hybrid dominance optimization

4.1 Motivation

In this final chapter on filter feature selection using dominance, we continue using microarray data. Cancer is one of the leading causes of death worldwide, affecting most families, and this work focuses on selecting a minimum optimal number of genes, which can lead to a great improvement in the classification accuracy of detecting cancer samples. Our proposed algorithm, improved Relevance-Redundancy Dominance with a Genetic Algorithm (iRRD-GA), uses an improved version of RRD (see Section 2.5.2) which was a univariate filter feature selection method. Based on dominance between both feature-to-feature and feature-to-target, this method reduces the dataset by removing dominated, irrelevant and redundant features. It selects a subset of features automatically, which are then further reduced by finding relationships between the subset of features using a dominant tournament style genetic algorithm (GA) where the dominant chromosomes are selected. This part of the feature selection is a wrapper, which due to iRRD selecting a small subset of features, can be implement efficiently. We demonstrate the effectiveness of iRRD paired with GA by analysing 24 microarray datasets, and achieving state-of-the-art results. We further experiment using 10 different and distinct classifiers, including Logistic regression, Ada-boost trees and multi-layer perceptron, which we believe have

not yet been fully researched in this area, both for the fitness function of the GA, and test classification accuracy. Our results show that once iRRD-GA selects a *good* subset of relevant informative genes, the selection of classifier becomes less important, as there is little variance between the results of the analysed classifiers. The experimental results are compared and contrasted with related work with similar cross-validation designs for a fair comparison.

4.2 Introduction

In 2018 there was about 17 million new cases of cancer worldwide, and about 9.6 million deaths from cancer. These statistics are startling, and any research that can help reduce these will be significant. In recent years, advancements in the technology biology scientists use to analyse DNA samples have improved greatly. The ability to search through thousands of gene expressions simultaneously helps to assist researchers in the detection of cancer tumours, or in the area of machine learning, accurately predict classes in a classification problem. A standard machine learning approach involves training a classifier to extract important information and patterns from the training data, which is then used to infer this information on a test set, hopefully, accurately classifying the unseen samples in their true categories. The characteristics of microarray data is such that it suffers from a well-known problem called "*curse of dimensionality*" [Bel55], which refers to when the dataset has too many features. One potential issue that this can cause is when there are significantly less samples than there are features, like in the case of microarray data which is usually several orders of magnitude lower, there is a higher risk of massively overfitting a prediction model, hence, generally resulting in a terrible out of sample performance.

Feature selection is commonly used in classification problems. It helps not only to reduce the complexity of the task, but also often ensures a better classification accuracy, by identifying the features most relevant and non-redundant to the target. In microarray analysis, feature selection is more commonly referred to as gene selection. The feature selection task can be defined as: given a dataset with features $F = \{f_1, ..., f_{n-1}, f_n\}$, retain a subset $F' \subseteq F$ that results in best classification accuracy function $\Theta : \Gamma \rightarrow G$ such that

$$F' = argmax_{G \subset \Gamma}\{\Theta(G)\} \tag{4.1}$$

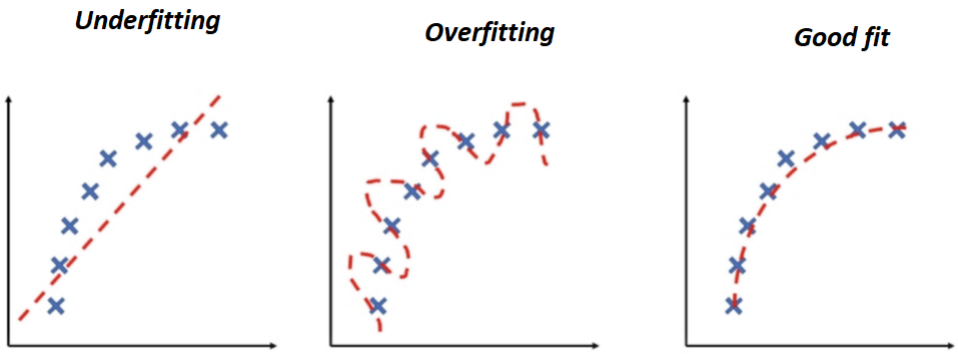

Figure 4.1: This is an example of a model underfitting and overfitting the data, and also shows a good fit of the data.

where Γ is the search space of all possible feature combinations of F and G a subset of Γ [TJGNA08]. The main reasons F' results in the best out of sample performance is that it reduces the possibility of overfitting on unseen data and it also reduces the complexity of the model. On high dimensional data an important part of pre-processing is finding a good selection of features, and as seen in Figure 4.1, getting the balance of just the right features can have a dramatic impact on the models out of sample predictions. If not enough informative features are selected the model will underfit, while if too many features are retained the model can overfit as some of the retained features can contain noise within them. The graph on the right of Figure 4.1 shows when a good subset F' is selected, it can help the model to improve it's predictive power. Although there are a large number of proposed methods, selecting the minimum optimal subset of genes for accurate classification still remains a challenging NP-Hard problem [NF77]. It has been shown that using naturally inspired evolutionary algorithms, based on Charles Darwin's evolutionary theory, can reduce the complexity of solving the problem. Using a filter feature selection, as our proposed method iRRD focuses on, decreases the task significantly to begin with.

Due to the large number of features the most popular method for gene selection is the filter method. A key difference that defines a filter method is that it does not use a learning model in the selection of features, instead relies on the statistical properties of the data. A filter method can be either uni-variate or multivariate [LRvVW06]. The univariate way calculates the correlation of each individual gene with respect to the target, and then ranks them most correlated to least. A predefined number of number of features are then selected, becoming the new subset representing the dataset. Usual determining criteria for the correlation are either mutual information, information gain and F-

score [CHYW09]. The main advantages of univariate methods are that they are fast, simple and efficient, but a drawback to them is they can have lower classification accuracy. This can be caused by not taking the relationship between selected features into account, hence the possibility of retaining highly correlated features. The multivariate filter method, however, overcomes the issue of multi-collinearity by taking into account the correlation between features. It sacrifices slight efficiency for a better classification accuracy, and regularly used examples in this area are mRMR [PLD05b] and FCBF [YL03b]. Some work have used an ensemble of filter methods to further improve their performance [SPBCAB16], [BCSMAB14a].

The next feature selection method is the wrapper method [JKP94]. This approach general achieves better prediction results compared to filter approaches, as it not only takes into account the relationship between features, similar to the multivariate filter method, but also uses a classifier to evaluate the selected subset of features. Different subsets are determined iteratively, with the ultimate aim of finally retaining the optimal subset. A wrapper approach can use either a greedy or stochastic search pattern. The greedy search pattern adds or removes individual features depending if it is step forward or step backwards feature selection. The main pitfall of this type of search is that it can easily get stuck in a local optimum. Whereas a stochastic search randomly select subsets, uses a classifier to evaluate their fitness, and then based on these results generate new, generally better, subsets, finally ending with an optimal subset. Probably one of the most popular stochastic search methods are the genetic algorithms (GA) [[SK07] [LCL$^+$05]], but others that have shown promise in this and various research areas include particle swarm optimization (PSO) [[LWT08] [SM12]] and ant colony optimization (ACO) [[TNRM15] [YGL$^+$09]]. Although, wrapper methods generally achieve greater classification results, due to the high dimensionality of microarray datasets the classification problem can be considered too computational expensive.

The final feature selection method is the embedded method. This method uses a classifier similar to the wrapper methods, but can learn which features can be best attributed to having a positive effect on the classifiers accuracy. This can be associated with the classifier having a greater impact on which features are selected. Random forest is an example of such an algorithm, but more commonly used in this area is support vector machine using recursive feature elimination (SVM-RFE) [MR09]. Again, same as the wrapper methods, the high dimensionality of the data makes this method extremely computationally

expensive.

Some research combines the idea of both the filter and wrapper methods to gain the best values out for both. These methods are sometimes called hybrid approaches, and this work follows along these lines [MDI05], [Jin14] and [AG13]. One of the best advantages of the filter method is its efficiency, while this is one of the main pitfalls in the wrapper methods. If a filter approach can be used to select a good small subset of relevant features, this would help to significantly reduce the computational complexity of the problem for the wrapper method. And since the wrapper method generally achieves better accuracy then the filter method, it can be used to determine the final optimal subset for the classification task. Stochastic search techniques have been shown to be a strong approach in this area, but possibly more important is the subset selected by the filter method, as this will be the only features in the exploratory search space.

In this research, we will adopt the hybrid approach to select the minimum optimal subset of genes to perform state-of-the-art classification accuracy. For the wrapper part we will use a genetic algorithm, which can be compared to various related work, but will introduce a novel filter method called *improved Relevance-Redundancy Dominance (iRRD)*. Unlike other filter methods, RRD automatically selects the optimal number of features to retain. Extensive experiments, using both binary and multi-class datasets, were carried out under varies cross-validation methods showing the performance and robustness of our proposed filter technique.

The main contributions of this chapter are as follows. We introduce a novel approach for selecting an optimal minimum subset of genes, which can be used by a genetic algorithm to achieve state-of-the-art results in this area. The experiments use both binary and multi-class datasets, thus showing the robustness of our proposed method. Finally, as far as we know this is the first to demonstrate the effectiveness of using feature dominance to select a good subset of genes to be used in a genetic algorithm.

This rest of this chapter is organised as follows, Section 4.3 discusses the related work on gene selection in microarray data. The datasets descriptions and classifiers used are given in Section 4.4, while we explain our proposed method in Section 4.5. The results are given in Section 4.6, as well as their evaluation and discussions. Finally, we conclude in Section 4.7.

4.3 Related Work

[BCSMAB15] presented a novel approach to perform feature selection, which involved a distributed method. Although the distributed part is unrelated to our proposed method, they also use various statistical and information theory based algorithms to select the subset of features within each partition and then combined them into a set of joint features as the final optimal subset. To improve the performance of their proposed method, they created the partitions using randomly selected features and using similarly ranked features. By using features that were similar, there would be a greater chance of them being removed during the implementation of the filter algorithm. The authors used the same microarray datasets and also classifiers as this work, and therefore their results can be compared to the results in this chapter. [VJR+19] follow a similar hybrid method, and showed how well an information theory correlation can be used in both ranking and filtering of genes, which we agree with.

[TDE18] compared 2 feature selection Strategies Recursive Feature Elimination (RFE) and Randomized Logistic Regression (RLR) using seven different classifiers including SVM, KNN, Ada-boost and Gradient Boosting Machines (GBM). Both feature selection methods required a predefined number of features to retain, which was set at 50. This could be considered a drawback to the feature selection methods, which, we will show, our proposed algorithm overcomes. Turgut's work will be a good comparison as they use a range of different classifiers, albeit, only on the breast cancer dataset. We use a similar cross-validation method, along with the same classifiers, and hence both works can be directly compared.

[MM16] uses a hybrid feature selection strategy, first rank the features by combining their Bayesian logistic regression, T-tests and Fisher methods results. Then, selecting the n top ranked features to be the subset which the PSO algorithm paired with discriminant independent component analysis (dICA) extracts a set number of important features. In this method, the n number of features from the filter method must be defined, along with the number of features to be extract using PSO-dICA. Although the wrapper version of this hybrid method differs from ours, we can still compare the results on the same experimental design and datasets, showing out proposed method automatically selects an initial better subset using iRRD. This in turn helps us to achieve a more accurate performance.

[LCY+17] opts to use mutual information maximization (MIM) as the filter feature selection technique, and an adaptive genetic algorithm (AGA) as the wrapper part of their hybrid method. We compare our results to the authors' and show that iRRD selects less features, but which are more informative as out results achieve better accuracy using SVM as the classifier, but also when compared with the other classifiers used by the author.

[DPHC18] demonstrates how a randomized wrapper can effectively select a "good" subset of features. In comparison to our work, they use a Pareto optimal based multi-objective genetic algorithm (PMOGA), but in contrast, they do not use a filter features selection to reduce the size of the set of features the GA can choose from. This increases the complexity of the problem, and hence we believe our proposed method is more efficient. They use a single classifier, radial basis function SVM, to measure the performance of their proposed method on 6 popular microarray datasets using a 10-fold validation technique. We set up the same experimental design, with the resulting analysis below.

[LLCK11] presented a novel adaptive GA evaluated using a KNN classifier to select genes and perform classification. The adaptive GA involved adapting the probability of the crossover and mutation, along with an elite strategy. The method was tested on the Colon dataset and showed to be promising, both in accuracy and efficiency, reinforcing using a GA for final gene selection.

[GBS+19] used a ranking of genes method followed by a genetic algorithm. The novel idea in their method is that the search space of the ranked genes decreased as the GA reached an optimum solution. The authors used SVM, KNN and MLP classifiers to evaluate the robustness of their proposed method on 5 binary and 2 multi-class microarray datasets. Apolloni et al, Zhang et al and Bonilla et al all also used a ranker filter feature selection before a type of GA on a range of datasets, and using various classifiers. All confirmed the effectiveness of the approach, reporting very competitive results.

[ABA15b, ABA15a] designed a Genetic Bee Colony (GBC), which is a variation of an ABC, using mRMR as a pre-processing step to eliminate noisy and redundant genes. GBC was compared to mRMR with an evolutionary algorithm, ABC without mRMR and mRMR with ABC. Again, their research illustrated that the use of a filter method to reduce the problem complexity followed by a evolutionary algorithm achieved the best subset of genes resulting in state-of-the-art accuracy.

More research work which employed a hybrid approach, showing that combining both a filter feature selection method for model efficiency with a randomized wrapper method for improved performance, include [HDH06], [SANA12], [YCY+10]. Others evaluated stand-only evolutionary algorithms, including [TJGNA08] who compared PSO to GA, and concluded the GA approach obtained better best solutions.

4.4 Classifiers and Datasets

4.4.1 Microarray Datasets

Table 4.1 gives a description of the 24 microarray datasets used in this research. The name of the dataset is followed by the number of classes, samples and genes/features. We also show the year of the dataset along with its original reference. To show the range of difficulty between the datasets, we show the number of samples within each class and its corresponding maximum imbalance ratio; the ratio between the class with the most samples and the class with the least samples. The number of classes is shown which defines the classification problem into either binary or multi-class. There were 18 binary datasets and 6 multi-class datasets, with the number of targets ranging from $3-5$. The number of genes in a dataset ranges from $2000-24481$, and the number of instances range from $47-253$, giving microarray data the rare property of having high-dimensionality with a very limited number of samples. This property is one of the main reasons microarray classification is a challenging task. Another reason is the imbalance ratio in some of the datasets, such as Lung4 where it is as high as $23:1$, but even an imbalance of anything over $3:1$ can be difficult (11 of the datasets have an imbalance greater than that).

4.4.2 Classifiers

An extensive range of experiments were carried out in this research work on microarray data, which involved 10 different by widely used classifiers. They use quite distinctive methods of classification, some considered to be more powerful than others, and some are a lot more efficient than others. Our results show that if the right subset of features are retained, then the job of the classifier is simplified and the classification accuracy across different classifiers are very similar. We use 3 tree-based classifiers; decision tree (DT), random forest (RF) and adaptive boosting (ada-boost) trees (ADA), $(1, 3 \,\&\, 5)$-k nearest neigh-

Table 4.1: A description of the microarray datasets used in this research work.

Dataset Name	#Classes	#Instances	#Features	#Instances per Class	Imbalance Ratio	Original Ref.
Brain	2	85	22283	59;26	2.27	[FCVF+04]
Bone	2	173	12625	137;36	3.81	[TZW+03]
Breast	2	95	24481	51;44	1.16	[VVDVDV+02]
CNS	2	60	7129	39;21	1.86	[PTG+02]
Colon	2	62	2000	40;22	1.82	[ABN+99]
Leukemia1	2	128	12625	95;33	2.88	[CLG+04]
Leukemia2	2	72	7129	47;25	1.88	[GST+99]
Leukemia3	3	72	7129	38;25;9	4.22	[ASS+02]
Leukemia4	4	72	7129	38;21;9;4	9.5	[GST+99]
Leukemia5	2	72	5147	47;25	1.88	[GST+99]
Leukemia6	3	72	12582	28;24;20	1.4	[GST+99]
Lung1	2	96	7129	86;10	8.6	[BKH+02]
Lung2	2	181	12533	150;31	4.84	[GJH+02]
Lung3	2	187	19993	97;90	1.08	[SBS+07]
Lung4	5	203	12600	139;21;20;17;6	23.17	[BRS+01]
Lymphoma1	2	47	4026	24;23	1.04	[AED+00]
Lymphoma2	2	77	7070	58;19	3.05	[SRT+02]
Lymphoma3	2	77	5469	58;19	3.05	[SRT+02]
Ovarian	2	253	15154	162;91	1.78	[PIAH+02]
Prostate1	2	102	10509	52;50	1.04	[SFR+02]
Prostate2	2	102	12600	52;50	1.04	[SFR+02]
Prostate3	2	136	12600	77;59	1.31	[SFR+02]
Prostate4	2	102	12533	52;50	1.04	[SFR+02]
SRBCT	4	83	2308	29;25;18;11	2.64	[KWR+01]

bours (KNN), logistic regression (LR), naive Bayes (NB), support vector machines (SVM) and multi-layer perceptron (MLP). Classifiers, logistic regression and SVM, are used for binary classification so in a multi-class task we use the common approach of One−Vs−All. We give below a brief explanation of adaboost trees and MLP, as the other classifiers have been explained (see Sections 2.4.4 and 3.4.2).

- ***multi-layer perceptron*** [Ros61] is a deep artificial neural network classifier, made up of many single perceptrons over a number of layers, shown in Figure 4.2. A perceptron typically takes a vector, x, multiples it by weights w and adds a bias b, giving an output, y. This can be generalized for multiple samples as seen in Eqn. 4.2, with an added non-linear activation function ϕ. MLP have an arbitrary number of layers between the input and output layer, all fully-connected. The weights and biases are adjusted iteratively, in order to minimize the error between the predicted output and the ground truth. The backward pass of the MLP model using back-propagation and the calculus chain rule, giving a gradient along

which the weights and biases are adjust to reduce the error. The gradient is found through differentiation of the partial derivatives of the error function with respect to the model parameters.

$$y = \phi\left(\sum_{i=1}^{N} w_i x_i + b\right) = \phi\left(w^T x + b\right) \tag{4.2}$$

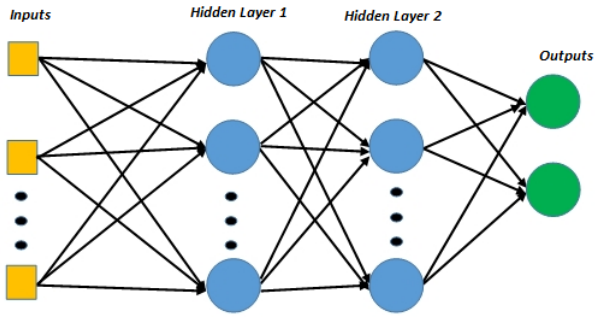

Figure 4.2: The figure shows a multi-layer perceptron, with n perceptrons per layer, across 2 hidden layers.

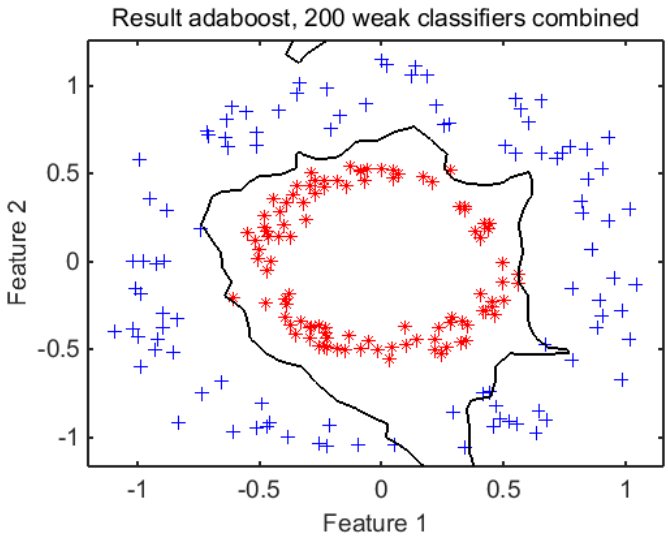

Figure 4.3: Adaptive boosting trees example, showing the boundary it learnt, through iteration, to separate classes. It trains a sequence of models with augmented sample weights, which helps to generate better parameter values for individual classifiers based on errors, thus finding a good decision boundary for generalization.

- ***Adaptive boosting trees***, also known as Ada-boost trees, [FS+96] is an ensemble boosting classifier, combining multiple classifiers in order to increase accuracy. It is an iterative approach that creates a strong classifier by combining weaker classifiers, such as decision trees, and adjusting assigned weights iteratively. It assigns each sample an initial weight value

of the multiplicative inverse of the number of samples, and then trains a decision tree. If the classifier correctly predicts a sample, its weight is decreased, and vice-versa, if the sample is wrongly classified, its corresponding weight is increased. A new decision tree is trained, giving more priority to the samples with larger weights. These steps are repeated until either the classifier predicts all the samples perfectly or that it has reached a user-defined maximum number of trees to attempt, as shown in Figure 4.3. Ada-boost trees are not susceptible to over-fitting, but are prone to noise within the data due to that it tries to fit each point perfectly. As it is an iterative process and needs adjustment of weights assigned to all points, it is the slowest classifier out of the 10.

4.5 Material and Methods

4.5.1 Experimental Design

The selected dataset is split into the standard training, validation and test sets, in accordance with the experimental design cross-validation. In microarray datasets the genes are usually continuous and, as research has shown, performance increases when discretization is carried out which helps smooth out the fluctuations in parts of the data through binning. Using the training dataset, we find the best *bins* for each gene, and then transform both the validation and test sets. To compare with the literature, we carry out experiments in 3 different cross-validation setups, 5-fold, 10-fold and Leave-one-out (LOOCV). In 3, 5 and 10 fold validation, the dataset is divided into relatively equal 3, 5 and 10 parts respectively, with 1 part for the test set, 1 part for the validation and the remaining sets as the training sets. The order of the selection of the parts are sequentially looped, so that each part has a turn of being a test, validation and training set. In LOOCV, each sample in turn, is assigned as the test-set, with a neighbouring one as the validation set. Therefore, the test set and validation set only every contain 1 sample, with the rest of the dataset used as the training set. An example of a k-fold of a 2 class task is shown in Figure 3.4.

4.5.2 Improved Relevance-Redundancy Dominance (iRRD)

In this section, we introduce the proposed iRRD-GA algorithm which efficiently selects informative genes from the microarray data. By selecting a minimum number of relevant genes, we obtain an improvement of the classification ac-

curacy of the classifier, along with improving its robustness due to reducing the possibility of over-fitting.

The main purpose of iRRD is to reduce the microarray dataset to a small subset of relevant and informative genes. This greatly helps to decrease the computational complexity of the genetic algorithm in order to find an optimal minimum subset of genes. Hence, by applying iRRD before implementing the GA, the search space is reduced, thus increases the GA ability to find a good solution. Unlike the original RRD (see Section 2.5.2) and FRRD (see Section 3.5.2, our proposed improved version iRRD (see Section 4) does not require a user input for a statistic, instead uses 2 defined statistics, Mutual Information I and Cramer's ϕ_c. For both statistics, iRRD ranks all the features in accordance between feature and target, sorted by most relevant being ranked top. Then, iRRD computes the statistic between each remaining feature and the current top feature, as well as the statistic between each remaining feature and the target, which is used to calculate β, the dominance ratio.

If the dominance ratio is less than or equal to β, we say it is dominated and the feature is removed, similar to the definitions stated in Section 3.5.2. It should be noted that for the results shown in this chapter β is set as 1, but we carried out a number of experiments to test the impact of varying this value. For the microarray datasets, evaluated in this chapter, setting β to 1 gave the best results but having the option to adjust the dominance ratio as a parameter could help achieve better classification accuracy on other experiments. Once the current top feature has removed all the features it dominates, it is put into a retained subset S. The process is repeated for the remaining top features until none remain.

We show that using 2 distinctive statistics, one based on information theory and the other on chi-squared, $\tilde{\chi}^2$, iRRD subsets the high-dimensional dataset into a small set of statistically un-dominated features. By using these 2 statistics, we get features selected on very different fields of view. Once iRRD has refined the original dataset into 2 much smaller subsets, it gets the union set D of both sets S. Unlike FRRD, where the user defined a number of features to retain, iRRD is an unsupervised approach, and from the results below, not only very accurate but due to the variance in D because of both statistics, only requires an extremely small set of features to achieve this accuracy. This subset D becomes the input of features for our Genetic Algorithm (GA). The time complexity of iRRD is of the order *O(n $log(n)$)*, where n represents the dimension

of the dataset.

The main difference between this filter feature selection part of the proposed approach, iRRD, is that instead of the user selecting which statistic to use, the algorithm has 2 defined statistics to use. This makes it a dual statistic method unlike the earlier algorithms RRD and FRRD (see Sections 2.5.2 and 3.5.2). Also, iRRD has a dominance ratio value which can help to control which features are added to the final output subset D, based on how much the feature is dominated. We believe this helps to add some diversity into the subset, which is expected to make it generalize better (perform better on unseen test samples). Unlike FRRD, it does not require the user to select how many features to retain, and unlike RRD, it uses 2 statistics rather than a single one selected by the user.

Algorithm 4 iRRD Feature Selection Algorithm

1: Set $\beta \leftarrow 1.0$
2: $D \leftarrow \emptyset$
3: **for all** $s \in [I\ ,\ \phi_c]$ **do**
4: given features F and target t
5: **for all** $f \in F$ **do**
6: $x_{ft} \leftarrow s(f, t)$
7: **end for**
8: $S \leftarrow \emptyset$
9: **while** $F \neq \emptyset$ **do**
10: $\hat{f} \leftarrow argmax_{f \in F} x_{ft}$
11: **for all** $f \in F \setminus \hat{f}$ **do**
12: **if** $\frac{x_{ft}}{s(\hat{f}, f)} \leq \beta$ **then**
13: $F \leftarrow F \setminus f$
14: **end if**
15: **end for**
16: $S \leftarrow S \cup \{\hat{f}\}$
17: **end while**
18: $D \leftarrow D \cup S$
19: **end for**
20: **return** D

The following bullet points give a line-by-line description of Algorithm 4:

- In line 1 we initialize the dominance ratio β to 1.0, which represents full domination of a feature.
- Line 2 is where we initialize D as an empty set, which we will fill with the features selected from both statistics, Mutual Information and Cramer's ϕ_c.

- Lines $3-19$ are a loop which is exited (line 19) when both statistics are used.
- This loop is the main part of the algorithm, and is where the most relevant features are determined and stored when analyzed by both statistics.
- In line 4 we define the features F and the target t from the dataset.
- Lines $5-7$ are a loop which is exited (line 7) when the all the features of the dataset have been analyzed.
- The purpose of this loop is to calculate both of the statistical values between each feature and the target, each statistic is calculated in turn.
- Line 8 is where we initialize S as an empty set, which we will fill with the selected features form both statistics.
- Lines $9-17$ are a loop which is exited (line 17) when the set of features F becomes a null set.
- This loop selects the important features, and removes groups of the dominated features.
- Line 10 selects the feature ($\hat{f}$) which has the greatest relevance, with respect to the target.
- Lines $11-15$ are a loop which is exited (line 15) when all the features of the set F have been analyzed, excluding the current most relevant feature.
- This loop removes the features which are dominated, with respect to the dominance ratio.
- Lines $12-14$ are an if condition; this checks the ratio between the statistical value of the remaining features and current most relevant feature, and the statistical value between the remaining features and the target. If the ratio is less than or equal to the dominance ratio β, the feature is deemed to be dominated and is removed.
- In line 16 we insert the current most relevant feature ($\hat{f}$) into the pre-initialized set S.
- Line 18 is where the new set of features S is unified with the pre-initialized set D.
- Line 20 ends the algorithm and returns the selected set of feature D.

4.5.3 Genetic Algorithm (GA)

Genetic algorithms are stochastic search methods based on biological natural selection evolution. We first explain 3 parts of the algorithm:

- ***Selection:*** By randomly picking 2 parent solutions, and selecting the most dominant one based on their fitness, ensures their good genes are passed to the next generation. This is repeated for the second parent. Hence, parents with a good fitness are more likely to be selected for breeding.
- ***Crossover:*** Crossover is an operation which splits 2 parent solutions and, by selecting one part from each parent, create 2 new child solutions. We perform one-point crossover, which allows combination information to pass from parent solution to child solution, thus retaining possible 'good' gene sequence.
- ***Mutation:*** Like in evolution, mutation can occur in particular new child solutions, and happens with a low random probability. The mutation flips genes to their current state opposite, which is important to maintain diversity within the population, allowing us to explore further feature space. This idea helps to prevent early convergence to a local optimum.

Our Genetic Algorithm (GA) initially creates a random population of size, three times the length of the chromosome (which is the size of the subset D selected by iRRD). Each chromosome is then evaluated, and assigned a fitness, using the classifier. The best fitness is recorded, along with the genes that achieved this result. A new evolved population is created from this older population, which then becomes extinct. The new population is the same size as the old one, and uses the individuals within the old population to create this evolved population. By randomly picking 2 pairs of parents, and then selecting the dominant parent in each pair, we end up with 2 good parent chromosomes. These 2 parents breed through one-point crossover creating 2 child chromosomes. By selecting the dominant parents each time, we ensure good gene sequences are passed onto the next generation, but by selecting the pairs randomly, it gives all chromosomes a chance to breed. We also use mutation as a technique to add random noise, which allows GA to explore a wider search space, hence helping to prevent GA getting stuck at a local optimum. Since we are not only interested in best performance, but also by obtaining it through the minimum number of relevant genes, we track also the number of genes associated with the best fitness. If we equal the best fitness, we next check if a reduced number of genes

were required, and if so update as best solution. The pseudocode for GA is shown in Algorithm 5 and a diagram of it in Figure 4.4. The time complexity of GA is of the order *O(gnm)*, where g is the number of generations, n being the population size and m the individual size. Combining the time complexity of GA and iRRD we get an overall time complexity of the order $O(n\ log(n) + gnm)$

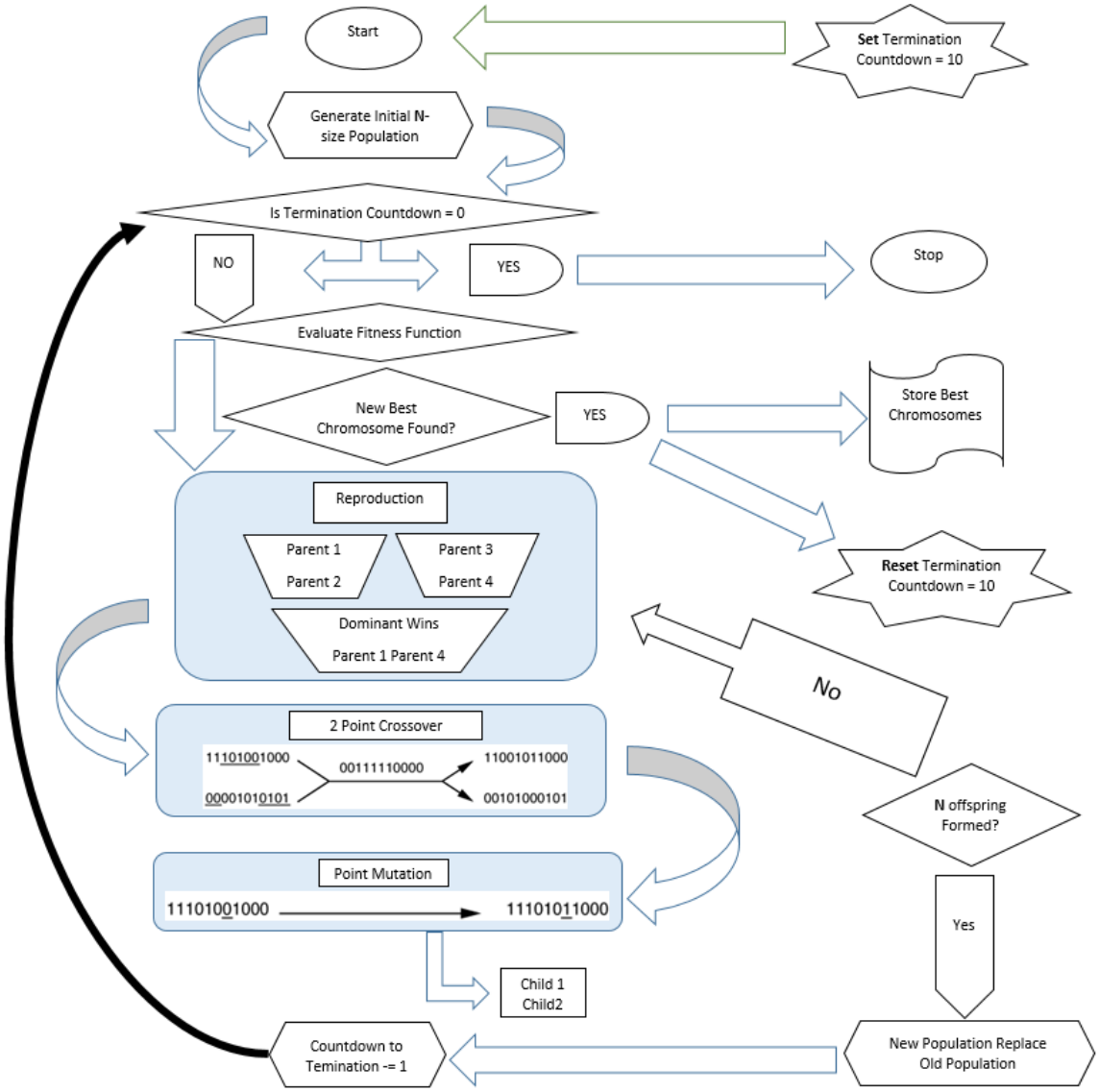

Figure 4.4: A diagram showing the flow of the proposed Genetic Algorithm.

4.6 Results

To compare the robustness of iRRD-GA, we run experiments using 4 different types of cross-validation; 3-fold, 5-fold, 10-fold and leave-one-out. This also help use to fairly compare and contrast with related work results, as much work

Algorithm 5 The pseudocode for the proposed Genetic Algorithm.

```
N ← population size
P ← create parent population by randomly creating N individuals
While termination condition not met
    C ← create empty child population
    While |C| < N
        parent1 ← randomly select individual
        parent2 ← randomly select individual
        parentX ← winner(parent1, parent2)
        parent3 ← randomly select individual
        parent4 ← randomly select individual
        parentY ← winner(parent3, parent4)
        child1, child2 ← crossover(parentX, parentY)
        mutate(child1) & mutate(child2)
        evaluate(child1) & evaluate(child2) for fitness
        insert(child1) & insert(child2) into C
    EndWhile
    P ← C
EndWhile
```

in this area uses one of these cross-validation setups. We also experiment on a large range of different and distinct microarray datasets, which the 24 datasets are described in Table 4.1. Much work has been carried out on microarray data, and various authors have shown results using a range of classifiers. Our work, will not only show results using 10 different classifiers (explained in Section 4.4.2), but our overall results show that if the right subset of informative features (genes) are selected then there is no significant difference between which classifier is used for the prediction. To show the impact iRRD-GA has on the model prediction, we compare each classifier accuracy, using no feature selection (which is the baseline), using only the features selected by step 1 in the algorithm iRRD and finally the features of our proposed approach iRRD-GA.

As can be seen in Table 4.2, some models overfitted on the full dataset badly (the large number of features in the datasets causing the classifiers not to generalize on out-of-sample data well, see Section 4.2); the 3 KNN methods on the breast dataset, while on the Leukemia1, Ovarian, SRBCT, Lung1 and Lung2 datasets a few of the classifiers were able to accurately classify all the test samples. The average result of the classifiers shows us that using the full dataset changes the prediction accuracy each classifier. Using a KNN or a Naive Bayes classifiers, only averages about 80%, while a random forest or logistic regres-

sion model can help score about 92% accuracy. The logistic regression model is rarely seen in the research area, but shows it is the overall most robust classifier on these set of experiments using the datasets described in Section 4.4.1. The 3-fold cross-validation results on the features selected by iRRD, Table 4.3, shows that the initial step of iRRD-GA has increased classification accuracy on some datasets, while decreasing it on others. Specifically, the weaker algorithms; Naive Bayes and KNN have had a $4\% - 5\%$ boost in accuracy and logistic regression and random forest decreased by about 2%. The average accuracy between all 10 classifiers has grouped closer together, with only a 5% difference compared previously being approximately 12%. It should be remembered that, to help to give the Genetic Algorithm (GA) more of a variant search space, we used 2 quite different statistics to choose the subset for iRRD, Mutual Information and Cramer's ϕ, the former based on information theory and the latter using chi-squared, $\tilde{\chi}^2$. This approach resulted in a combination of unique features from 2 different points of view of the data. This meant that, although we selected a reasonably small subset of relevant features on average about 25, due to the unification of both sets of features selected by the pair of statistics the subset could contain feature correlations or dominated features. Hence, this can cause a prediction reduction but, as well documented, with a significantly reduced number of features classification models generalize better. The final set of results for 3-fold cross-validation, Table 4.4, shows that the informative subset of features iRRD-GA selected were able to achieve state-of-the-art results across all datasets using all classifiers, and in a lot of cases scoring with 100% accuracy. Our proposed method, iRRD-GA, accomplished this using an extremely small subset of features, on average 4.5 across the classifiers. It can also be seen that the average accuracy across classifiers has very little variance, about 1%, which can indicate that once a *good* subset of features are selected as the input for a classifier, the classifier used is almost irrelevant. This is clearly seen where a simple KNN-1 model scores 96.53% using 3.95 features, while more complicated and computationally more expensive model — namely ada-boost tree, random forest and support vector machine — have slightly worse prediction accuracies while selecting slightly more features. A multi-layer perceptron only scores 0.05% better than the KNN model, but is vastly more complex, and hard for explainability. When we compare our work with the current state-of-the-art, using the same experimental design (Table 4.5), we see iRRD-GA has a better prediction accuracy on 8 out of the 12 results. In some experiments only scoring slightly better, while in other experiments having nearly 26% better classifica-

tion accuracy. Out of the 4 experiments in which iRRD-GA was beaten, 2 of them we had the same accuracy, but used 0.3 and 0.7 more features than the related work. Another related work had a 0.48% better accuracy, which is insignificant, but one surprise result was the Breast dataset, on which we were beaten by 13.68%, albeit using an average of 11.7 less features. Overall, on a 3-fold cross-validation design, our proposed method, iRRD-GA, can be considered a state-of-the-art hybrid feature selection approach. A full table of the related work results using 3-fold cross-validation can be found in the appendix (Tables A.1 and A.2).

Table 4.2: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 3-fold cross-validation design. These are the baseline results using no feature selection, hence all features are selected.

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	82.08	78.6	72.24	72.84	79.19	79.2	79.19	79.19	79.76	79.78
Brain	90.6	88.26	81.12	85.88	83.58	84.81	89.41	88.26	92.94	92.9
Breast	72.75	69.62	67.41	52.62	57.9	56.72	63.31	70.6	76.92	75.84
CNS	71.67	63.33	58.33	55.0	60.0	56.67	70.0	65.0	70.0	70.0
Colon	85.56	77.46	80.63	68.02	62.94	66.03	79.05	83.89	80.79	80.71
Leukemia1	100.0	100.0	95.35	94.48	96.11	95.31	100.0	98.45	100.0	99.22
Leukemia2	98.61	91.67	86.11	86.11	84.72	84.72	97.22	90.28	94.44	93.06
Leukemia3	97.22	93.06	88.89	80.56	83.33	79.17	93.06	84.72	97.22	94.44
Leukemia4	94.5	81.93	84.76	77.93	81.93	74.92	91.77	79.08	91.71	88.93
Leukemia5	98.61	93.06	98.61	88.89	87.5	88.89	97.22	98.61	94.44	91.67
Leukemia6	98.61	87.5	90.28	79.17	81.94	79.17	94.44	93.06	95.83	86.11
Lung1	100.0	100.0	98.96	100.0	98.96	100.0	100.0	100.0	98.96	100.0
Lung2	98.89	96.13	99.44	95.03	93.37	92.26	98.89	100.0	98.89	98.89
Lung3	79.66	70.59	62.57	62.02	64.68	65.76	72.7	67.92	77.0	74.36
Lung4	96.55	93.58	94.1	93.6	92.61	90.14	94.59	94.57	94.59	92.15
Lymphoma1	91.67	91.53	63.75	59.58	63.47	67.78	85.14	91.53	93.61	85.0
Lymphoma2	98.67	90.87	77.85	77.95	80.51	85.69	96.05	85.79	92.26	91.03
Lymphoma3	97.38	90.97	79.33	75.23	81.74	82.97	96.05	88.41	93.49	97.38
Ovarian	100.0	97.23	87.37	92.1	91.71	93.29	99.6	95.26	98.02	98.02
Prostate1	95.1	87.25	60.78	82.35	81.37	79.41	93.14	83.33	94.12	91.18
Prostate2	95.1	85.29	60.78	69.61	80.39	80.39	93.14	76.47	94.12	93.14
Prostate3	91.95	85.3	55.14	73.51	80.89	83.09	89.76	79.37	92.64	94.14
Prostate4	94.12	81.37	61.76	79.41	77.45	80.39	92.16	73.53	92.16	93.14
SRBCT	100.0	89.24	100.0	90.3	92.77	92.72	100.0	96.43	100.0	93.96
Average	92.89	86.83	79.4	78.84	80.79	80.81	90.25	85.99	91.41	89.79

As can be seen in Table 4.6, for the 5-fold cross-validation with no feature selection, the results follow the same pattern as 3-fold design. Naive Bayes and KNN

Table 4.3: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 3-fold cross-validation design. The classification results are from the features selected, from step 1 of the algorithm, iRRD, and the number of features selected are given in the table.

Dataset	#Feats	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	25.7	78.01	75.14	72.8	75.69	73.38	75.69	79.18	82.07	76.84	74.56
Brain	27.7	82.39	83.46	78.78	83.54	80.01	81.2	88.22	89.45	89.41	89.41
Breast	28	70.5	66.33	63.21	67.34	66.33	64.18	73.66	70.53	70.5	68.48
CNS	24	66.67	56.67	50.0	60.0	55.0	53.33	63.33	71.67	61.67	55.0
Colon	19	85.4	71.11	82.14	77.54	69.29	69.21	80.63	87.06	75.87	77.46
Leukemia1	25.3	100.0	100.0	100.0	99.22	99.22	99.22	100.0	100.0	100.0	99.22
Leukemia2	23.7	91.67	90.28	91.67	90.28	91.67	88.89	90.28	95.83	93.06	88.89
Leukemia3	26.3	97.22	94.44	94.44	91.67	94.44	91.67	94.44	97.22	95.83	94.44
Leukemia4	28.3	94.38	86.16	93.05	92.99	90.26	87.54	88.81	88.87	90.32	79.26
Leukemia5	23	94.44	90.28	90.28	90.28	91.67	90.28	91.67	95.83	90.28	91.67
Leukemia6	28.3	91.67	88.89	95.83	88.89	84.72	84.72	90.28	91.67	97.22	87.5
Lung1	24	100.0	97.92	98.96	98.96	98.96	98.96	100.0	98.96	98.96	97.92
Lung2	26	98.89	97.24	98.89	97.23	97.79	97.8	99.44	100.0	98.33	98.89
Lung3	27	69.53	68.95	64.7	57.77	65.22	68.45	70.58	68.44	70.03	63.63
Lung4	25	90.15	87.69	89.67	91.12	91.62	92.12	89.66	88.18	89.66	87.19
Lymphoma1	26	85.14	87.36	72.08	76.39	68.06	70.28	80.83	89.31	87.22	87.36
Lymphoma2	23.3	84.41	86.97	84.41	81.79	81.85	85.64	84.36	89.59	89.64	85.74
Lymphoma3	23.3	85.64	85.59	84.36	83.03	81.79	86.92	96.0	86.92	88.31	89.59
Ovarian	25.3	98.81	97.23	96.43	96.43	97.62	97.62	98.81	98.02	97.22	96.83
Prostate1	25.3	90.2	85.29	81.37	87.25	87.25	86.27	94.12	89.22	89.22	91.18
Prostate2	25.3	89.22	81.37	85.29	73.53	79.41	79.41	88.24	83.33	87.25	89.22
Prostate3	25.7	90.42	92.67	60.24	81.58	85.28	84.54	88.94	88.97	94.12	88.95
Prostate4	25.3	93.14	87.25	90.2	83.33	87.25	87.25	93.14	90.2	88.24	90.2
SRBCT	23	100.0	90.34	97.58	95.19	97.62	97.62	98.81	98.77	96.38	89.11
Average	25.16	88.66	85.36	84.02	84.21	83.99	84.12	88.48	89.17	88.15	85.9

models having the least predictive power, while once again logistic regression tops the accuracy results, with random forest, SVM and ada-boost trees close behind. Looking at the average classification across the classifiers, we see that there is a variance of 13%, showing when using the full high-dimension dataset the classifier selection is very important having a significant effect on the outcome. The 5-fold cross-validation results on the features selected by iRRD, Table 4.7, again are similar to the 3-fold setup, with the most accurate classifiers taking a small decrease in accuracy due to the refined subset of features, while the weaker classifiers improving in accuracy. The average accuracy between the 10 classifiers, on all the datasets, has less of a range of accuracies compared to when no feature selection is used, having only a 5% difference. Because of this closer cluster of results, due to less features being used for prediction, the classifiers are able to more accurately classify unseen test samples. Look-

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	87.86 (5)	86.12 (3)	84.96 (3.3)	86.7 (3.7)	85.53 (3.7)	87.86 (6)	87.28 (5)	86.69 (4.7)	87.26 (3.3)	87.83 (7.3)
Brain	97.66 (6)	95.32 (4.3)	97.66 (4.7)	98.85 (6.3)	97.66 (6)	95.28 (5.7)	97.66 (6)	100.0 (7)	96.47 (6.3)	97.62 (6)
Breast	82.09 (8.7)	84.24 (6)	82.06 (7)	82.09 (2.3)	84.17 (7)	85.25 (6.3)	83.13 (8.3)	85.25 (6)	84.17 (7)	86.32 (6.3)
CNS	85.0 (5)	90.0 (3.3)	80.0 (4.7)	90.0 (2.3)	83.33 (2.3)	85.0 (2)	83.33 (2.3)	88.33 (8.7)	86.67 (2.7)	90.0 (4.7)
Colon	91.98 (3)	90.32 (3.3)	91.98 (3.7)	96.83 (4.7)	93.57 (3.7)	91.98 (3.7)	90.32 (3)	96.83 (4)	93.65 (2.7)	95.24 (3.7)
Leukemia1	100.0 (1)	100.0 (1.3)	100.0 (2)	100.0 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia2	98.61 (4.7)	97.22 (4)	95.83 (3.7)	100.0 (3.7)	98.61 (3)	98.61 (5)	97.22 (2.3)	100.0 (3.3)	97.22 (2.3)	98.61 (3)
Leukemia3	100.0 (4)	98.61 (2.7)	100.0 (4.7)	100.0 (3)	100.0 (3.7)	100.0 (3)	100.0 (3)	100.0 (3)	98.61 (3)	100.0 (4)
Leukemia4	100.0 (5.7)	97.28 (4)	100.0 (8)	100.0 (5)	100.0 (7)	98.67 (4)	100.0 (5.7)	97.28 (4)	98.67 (5.7)	95.83 (3.3)
Leukemia5	100.0 (2.3)	98.61 (2)	100.0 (2.7)	100.0 (3)	100.0 (4)	98.61 (1.7)	100.0 (3)	100.0 (2.3)	100.0 (2.7)	100.0 (3.7)
Leukemia6	100.0 (4)	98.61 (3.3)	98.61 (5.3)	97.22 (4.3)	100.0 (5.3)	100.0 (8.3)	100.0 (5)	100.0 (5.3)	100.0 (3.7)	100.0 (9)
Lung1	100.0 (1.7)	100.0 (1)	100.0 (2.7)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1.3)	100.0 (1)	100.0 (1)
Lung2	100.0 (2)	100.0 (2.3)	100.0 (3.3)	100.0 (2)	100.0 (2.7)	100.0 (2.3)	100.0 (2)	100.0 (1.7)	100.0 (2.7)	100.0 (2.3)
Lung3	80.21 (6.3)	79.68 (6)	73.8 (5.3)	81.82 (5)	85.01 (6)	83.42 (7)	82.35 (5.7)	80.75 (5.3)	83.94 (10)	83.41 (6.7)
Lung4	96.05 (11)	93.6 (6.3)	95.08 (8)	97.04 (10.3)	97.03 (8.3)	96.05 (9.3)	99.01 (10)	92.62 (11)	94.58 (8.3)	95.57 (11.3)
Lymphoma1	97.92 (3.7)	93.75 (4)	97.92 (4.7)	97.92 (3.7)	97.92 (3.7)	95.83 (3.7)	97.92 (4.3)	100.0 (3)	91.53 (3.3)	93.75 (2.3)
Lymphoma2	98.72 (7.3)	94.77 (3.7)	94.77 (7.3)	98.72 (3)	100.0 (6.7)	100.0 (5)	98.72 (5.3)	98.72 (3.3)	97.44 (6)	98.72 (4.7)
Lymphoma3	94.77 (3.3)	94.77 (3.7)	93.44 (3.7)	98.67 (5)	97.33 (3.3)	97.33 (4.7)	98.67 (5.3)	97.33 (9.7)	96.05 (4.3)	96.05 (3.3)
Ovarian	100.0 (3.3)	99.21 (3)	99.6 (3.3)	100.0 (3.3)	100.0 (2.7)	99.6 (5)	100.0 (3)	100.0 (2.7)	99.6 (3.7)	100.0 (4.7)
Prostate1	98.04 (3.7)	96.08 (3.3)	97.06 (6)	97.06 (3)	98.04 (5)	98.04 (4)	99.02 (4.7)	99.02 (5.7)	97.06 (4)	99.02 (5)
Prostate2	98.04 (5.3)	94.12 (4.3)	96.08 (4.7)	98.04 (4.3)	95.1 (2.7)	96.08 (5)	99.02 (3.7)	98.04 (3.3)	95.1 (4)	97.06 (5)
Prostate3	97.05 (6)	96.31 (4.7)	94.12 (5.7)	97.79 (5)	98.52 (4.7)	98.52 (7.7)	97.05 (5.7)	97.79 (7)	97.79 (7.3)	97.78 (6.7)
Prostate4	97.06 (4.3)	97.06 (4)	96.08 (4)	98.04 (4.7)	98.04 (6.3)	98.04 (4.7)	98.04 (4.3)	99.02 (4.7)	98.04 (4.3)	99.02 (5.3)
SRBCT	100.0 (4)	100.0 (5.7)	100.0 (6.7)	100.0 (5)	100.0 (5)	100.0 (6)	100.0 (4.3)	100.0 (7.7)	100.0 (5)	100.0 (6.7)
Average	95.88 (4.64)	94.82 (3.72)	94.54 (4.8)	96.53 (3.95)	96.24 (4.37)	96.01 (4.67)	96.2 (4.33)	96.57 (4.82)	95.58 (4.35)	96.33 (4.88)

Table 4.4: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 3-fold cross-validation design. The table shows the classification accuracy for iRRD-GA, with the number of selected features in brackets, for each classifier.

Table 4.5: Comparison of the current State-of-the-art results and iRRD-GA using 3-fold cross-validation (best results shown in bold).

	Related Work				iRRD-GA	
Dataset	Reference	#Features	Accuracy	Year	#Features	Accuracy
Breast	[BCSMAB15]	18	**100**	2016	6.3	86.32
CNS	[SPBCAB16]	5	75.00	2016	2.3	**90.0**
Colon	[SPBCAB16]	25	85.00	2016	3.7	**95.24**
Leukemia2	[KSM15]	3	95.89	2015	3.3	**100**
Leukemia6	[SIM11]	4	**100**	2011	3.7	100
Lung2	[SPBCAB16]	5	99.33	2016	1.7	**100**
Lung4	[TNRM15]	20	85.72	2016	10	**99.01**
Lymphoma1	[SPBCAB16]	5	93.33	2016	3	**100**
Ovarian	[SPBCAB16]	2	**100**	2016	2.7	100
Prostate1	[TNRM15]	20	73.15	2016	4.7	**99.02**
Prostate3	[SIM11]	4	**100**	2011	4.7	98.52
SRBCT	[SIM11]	4	**100**	2011	4	**100**

ing at the results for iRRD-GA, Table 4.8, once again we see that our method achieves state-of-the-art results, in some cases being less than 2% off having a perfect prediction. On average iRDD-GA only uses 3.5 features to score this classification, hence it is not only accurate but also efficient. These results show that iRRD-GA creates robust classifiers, with the weakest classifier achieving 95.98% and the best achieving 98.37%, a variance of only 2.5%. A comparison between the current state-of-the-art and our approach, using the same experimental setup (Table 4.9), we see iRRD-GA has a better prediction accuracy on 17 out of the 18 results. On the Lung3 dataset, iRRD-GA has nearly 18% better classification accuracy. On over half of the datasets, iRRD-GA selects less than 2.5, and on the Lung3 dataset it requires 10.4 features to confidently beats the current state-of-the-art. On the ovarian dataset, iRRD-GA was beaten by 0.39% accuracy, but where [BCSMAB$^+$14c] used 31 features, our approach only used 2.2 features. A full table of the related work results using 5-fold cross-validation can be found in the appendix (Table A.3).

On the 10-fold experimental design, the baseline results using no feature selection, Table 4.10 shows the Naive Bayes model generalizes worst on the full dataset, with the KNN models only doing $2\% - 3\%$ better. Once again, the lesser used classifier logistic regression has the best classification accuracy with 94.43% outperforming the 4 computationally, very expensive classifiers multilayer perceptron, SVM, random forest and ada-boost tree. Table 4.11 shows

Table 4.6: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 5-fold cross-validation design. These are the baseline results using no feature selection, hence all features are selected.

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	83.8	76.91	75.14	75.75	78.62	79.21	81.48	79.19	81.51	80.35
Brain	92.94	83.53	81.18	91.76	81.18	85.88	91.76	92.94	91.76	90.59
Breast	80.0	64.21	69.47	55.79	54.74	53.68	67.37	72.63	76.84	71.58
CNS	80.0	60.0	61.67	50.0	63.33	60.0	71.67	75.0	76.67	73.33
Colon	90.51	80.77	85.64	77.44	71.03	72.56	84.1	90.64	88.85	85.64
Leukemia1	100.0	100.0	94.49	94.46	92.12	95.32	100.0	97.63	100.0	99.2
Leukemia2	98.57	90.29	90.0	87.62	86.19	87.52	98.57	98.57	93.05	93.05
Leukemia3	98.46	92.0	95.38	81.55	84.21	78.67	96.92	97.13	95.79	94.46
Leukemia4	92.95	93.05	86.1	84.57	78.95	78.95	91.52	81.9	91.62	91.62
Leukemia5	98.57	90.1	95.9	84.76	88.76	88.67	98.57	98.57	95.71	91.43
Leukemia6	98.67	91.79	93.24	88.0	84.0	88.0	96.0	91.79	97.13	97.13
Lung1	100.0	100.0	99.0	98.95	99.0	100.0	100.0	100.0	100.0	100.0
Lung2	98.9	97.24	99.44	96.7	93.38	92.84	98.9	99.44	99.44	98.36
Lung3	77.54	66.39	63.02	65.72	68.48	70.04	71.08	72.15	76.93	73.78
Lung4	95.54	95.56	93.11	93.07	92.07	90.58	94.05	93.58	95.57	88.19
Lymphoma1	89.33	89.56	68.44	70.22	67.78	71.56	85.11	89.11	96.0	94.0
Lymphoma2	97.5	82.83	77.75	80.75	89.67	88.42	96.17	79.25	86.92	93.5
Lymphoma3	98.75	90.92	80.58	84.58	87.0	88.42	97.5	92.25	94.92	94.83
Ovarian	100.0	98.42	87.36	94.47	94.45	92.9	99.61	96.05	98.43	99.61
Prostate1	96.05	83.24	61.62	86.33	83.24	86.19	92.05	86.19	93.05	92.1
Prostate2	94.1	85.24	60.9	75.43	83.33	81.43	90.14	79.38	95.1	91.19
Prostate3	91.93	85.26	55.08	77.25	80.13	78.7	91.93	81.69	90.48	93.41
Prostate4	94.19	86.24	61.95	84.48	83.43	86.43	90.33	86.29	93.14	91.24
SRBCT	100.0	91.62	100.0	92.72	93.9	92.65	100.0	96.32	100.0	98.75
Average	93.68	86.47	80.69	82.18	82.46	82.86	91.03	88.65	92.04	90.72

the accuracy of the classifiers on the subset selected by iRRD, where an average of 25.34 features were retained across all the datasets. This subset of features, although reduced the accuracy in a couple of classifiers, improved the accuracy of Naive Bayes, KNN and multi-layer perceptron. Our proposed method, Table 4.12, has a prediction percentage of over 99% on 8 of the 10 classifiers, with the other 2 scoring 97.5% and 98.5%. On the colon, lung1, lung2, lymphoma1, ovarian, SRBCT and all the leukemia datasets, iRRD-GA has perfect classification accuracy across all classifiers. Overall, iRRD-GA achieves these results using only about 2 features. A comparison between the current state-of-the-art and our approach, using the same experimental setup (Table 4.13), we see iRRD-GA has a better classification accuracy on all of the microarray datasets.

Table 4.7: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 5-fold cross-validation design. The classification results are from the features selected, from step 1 of the algorithm, iRRD, and the number of features selected are given in the table.

Dataset	#Feats	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	25.4	79.78	76.27	76.3	72.24	80.34	79.75	79.75	80.92	80.37	75.75
Brain	28	89.41	87.06	87.06	87.06	88.24	89.41	89.41	90.59	91.76	90.59
Breast	28	68.42	70.53	62.11	63.16	63.16	68.42	71.58	71.58	75.79	66.32
CNS	23.8	66.67	73.33	66.67	58.33	63.33	63.33	70.0	75.0	73.33	75.0
Colon	19.6	88.85	85.38	88.85	82.44	82.44	82.44	88.72	93.72	88.72	82.31
Leukemia1	25.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.23	100.0	99.2
Leukemia2	24.2	95.71	90.38	93.05	92.86	91.52	91.52	97.14	95.81	93.05	94.38
Leukemia3	23.4	92.92	88.83	92.92	87.7	91.49	86.07	94.26	89.14	94.46	92.92
Leukemia4	29	86.0	83.14	88.76	86.1	86.1	82.0	86.0	86.19	87.33	84.67
Leukemia5	23.8	91.62	90.19	97.24	88.86	90.19	90.19	97.24	94.48	91.52	88.86
Leukemia6	28.2	97.13	84.42	94.26	90.16	88.83	88.92	94.26	94.37	94.05	85.96
Lung1	24	100.0	100.0	99.0	99.0	99.0	99.0	99.0	98.95	100.0	100.0
Lung2	26	98.36	97.25	98.36	96.71	96.71	97.27	97.82	97.82	97.81	98.36
Lung3	27.6	68.41	67.85	61.44	64.08	70.5	66.74	68.89	71.66	72.13	71.02
Lung4	25.4	93.62	92.1	87.67	88.62	89.63	89.62	91.11	84.78	93.1	87.63
Lymphoma1	26.4	83.11	72.22	78.67	80.67	80.44	78.22	84.89	98.0	85.11	79.11
Lymphoma2	23.8	85.75	79.0	81.92	85.83	82.08	85.92	88.42	88.42	76.5	80.5
Lymphoma3	24	89.83	85.75	85.83	80.42	80.58	80.67	88.42	92.25	88.42	87.0
Ovarian	27.4	98.43	96.85	96.06	98.04	98.43	97.64	98.43	98.03	97.64	98.82
Prostate1	25.2	94.05	85.24	87.1	88.19	89.05	89.1	93.05	93.05	88.14	93.1
Prostate2	25	89.19	86.19	84.38	79.33	81.24	76.38	89.24	85.33	91.19	87.29
Prostate3	25.6	89.71	84.55	61.75	86.01	85.32	85.34	88.28	89.79	88.23	89.68
Prostate4	25.6	96.1	87.29	88.33	87.38	87.43	89.38	94.19	93.19	94.14	90.24
SRBCT	22	98.75	89.26	98.75	97.57	98.75	100.0	100.0	98.82	95.15	95.15
Average	25.29	89.24	85.54	85.69	85.03	86.03	85.72	89.59	90.05	89.08	87.24

On the Lung3 dataset, iRRD-GA has nearly 22.5% better classification accuracy, and on the bone, brain and leukemia4 datasets, our approach had about 15% greater accuracy. On 18 out of the 23 datasets, iRRD-GA selects less than 2 features, and on the Lung3, Lung4 and Bone dataset it selected about 5 features to outperform the current state-of-the-art. On all but 2 datasets, iRRD-GA selected a small subset of relevant and informative features to achieve perfect prediction scores, while on the other 2 datasets, still achieved state-of-the-art, scoring 96.26% and 98.86%. A full table of the related work results using 10-fold cross-validation can be found in the appendix (Tables A.4, A.5 and A.6).

The final cross-validation design under analysis is Leave-One-Out cross-validation (LOOCV), and due to the computational expensive both random forest and ada-boost tree classifiers are not explored in this section. The baseline results (see Table 4.14) show a slight difference from the other k-fold validation

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	86.74 (4.8)	87.88 (2.6)	85.58 (4.6)	91.9 (3)	90.76 (3.8)	91.34 (5.6)	88.45 (4)	87.29 (3.6)	91.92 (5)	91.92 (5.4)
Brain	98.82 (2.8)	98.82 (2.4)	98.82 (4.4)	100.0 (3)	98.82 (4)	98.82 (3.2)	100.0 (3.2)	100.0 (2.8)	98.82 (3)	98.82 (4.2)
Breast	85.26 (3.8)	90.53 (6.4)	88.42 (5.6)	91.58 (5.6)	90.53 (5.6)	92.63 (7.8)	89.47 (6)	91.58 (6)	91.58 (5.6)	89.47 (5.2)
CNS	90.0 (5.4)	95.0 (3.4)	83.33 (4.2)	95.0 (2.6)	98.33 (3.8)	93.33 (4.8)	93.33 (3.2)	95.0 (4.4)	95.0 (3.4)	96.67 (2.8)
Colon	96.79 (2)	96.79 (2)	95.26 (2.6)	100.0 (2.2)	98.46 (2.6)	95.26 (2.6)	96.79 (3)	100.0 (2.8)	100.0 (2.2)	96.92 (2.8)
Leukemia1	100.0 (1)	100.0 (1)	100.0 (2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia2	100.0 (1.2)	100.0 (1.6)	100.0 (2.8)	100.0 (1.6)	100.0 (2.2)	100.0 (2)	100.0 (1.4)	100.0 (1.2)	100.0 (2.2)	100.0 (1.8)
Leukemia3	100.0 (2.4)	100.0 (2.2)	100.0 (3.8)	100.0 (2.2)	100.0 (2.2)	100.0 (2.4)	100.0 (2.2)	100.0 (3.4)	100.0 (2.4)	100.0 (2.8)
Leukemia4	98.57 (4)	97.24 (3.4)	95.81 (4.6)	100.0 (3.8)	98.57 (3.6)	94.38 (3)	97.14 (2.8)	98.57 (4.2)	98.57 (3.4)	95.81 (2.8)
Leukemia5	100.0 (1.2)	100.0 (1.4)	100.0 (2)	100.0 (1)	100.0 (1.4)	100.0 (1.6)	100.0 (1)	100.0 (1)	100.0 (1.2)	100.0 (1.6)
Leukemia6	100.0 (2.8)	100.0 (2.8)	98.67 (4.2)	100.0 (3)	98.67 (2.6)	100.0 (4.4)	100.0 (3)	100.0 (4)	100.0 (3.4)	98.67 (2.6)
Lung1	100.0 (1)	100.0 (1)	100.0 (2)	100.0 (1.2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung2	98.9 (1.2)	100.0 (1.4)	99.44 (3)	100.0 (1.8)	100.0 (1.6)	100.0 (1.8)	100.0 (1.6)	100.0 (1.4)	99.46 (1.4)	100.0 (2.6)
Lung3	83.95 (8.2)	86.07 (6)	80.71 (4.8)	89.27 (8)	89.3 (5.4)	88.75 (5.8)	86.6 (7.2)	85.01 (5)	90.91 (7)	90.92 (10.4)
Lung4	99.02 (9.6)	98.54 (5.4)	98.05 (7.4)	97.56 (5.8)	98.05 (8.4)	97.07 (6.4)	99.52 (7.4)	93.61 (9)	98.01 (5.4)	95.57 (9.2)
Lymphoma1	100.0 (1.8)	98.0 (2.2)	97.78 (2.2)	100.0 (2)	100.0 (2.2)	100.0 (2.2)	100.0 (1.8)	100.0 (2)	100.0 (2)	100.0 (2.4)
Lymphoma2	98.75 (3.4)	98.75 (2.4)	97.5 (3)	98.75 (2.2)	98.75 (2)	100.0 (3.4)	98.75 (2.2)	100.0 (2.4)	100.0 (3.6)	98.75 (2.6)
Lymphoma3	97.42 (2.6)	98.75 (2)	97.42 (2.8)	100.0 (2)	100.0 (2)	100.0 (3.2)	98.75 (2.4)	100.0 (3)	100.0 (2.2)	98.75 (2.2)
Ovarian	99.61 (2.6)	99.22 (2.4)	99.22 (2.8)	99.61 (3)	99.22 (2.6)	99.22 (2.6)	99.61 (2.2)	99.61 (2.2)	99.22 (2.4)	99.61 (3.6)
Prostate1	99.0 (4)	99.0 (4.2)	96.05 (2.6)	100.0 (3.2)	99.0 (3.4)	99.0 (4)	99.0 (3.6)	100.0 (4.4)	99.0 (4.6)	99.0 (4.6)
Prostate2	97.05 (2.2)	98.05 (3)	97.05 (3.4)	97.1 (2)	98.05 (3.2)	98.05 (2.8)	98.05 (2.2)	98.05 (2.4)	98.05 (3)	98.05 (3)
Prostate3	97.8 (5)	97.09 (4)	96.32 (5.8)	100.0 (5.4)	100.0 (5.2)	99.26 (5)	97.09 (4.8)	99.26 (4.8)	99.26 (6.6)	97.06 (5)
Prostate4	99.05 (4.8)	99.0 (2)	98.05 (2.8)	100.0 (2.4)	99.05 (3)	99.05 (3.6)	98.1 (3.4)	100.0 (2.8)	99.0 (2.6)	100.0 (3.4)
SRBCT	100.0 (3.6)	100.0 (4.6)	100.0 (4.8)	100.0 (3.4)	100.0 (3.8)	100.0 (3.4)	100.0 (3.6)	100.0 (4.4)	100.0 (3.4)	100.0 (3.6)
Average	96.95 (3.39)	97.45 (2.91)	95.98 (3.67)	98.37 (2.98)	98.15 (3.19)	97.76 (3.48)	97.53 (3.09)	97.83 (3.3)	98.28 (3.25)	97.75 (3.61)

Table 4.8: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 5-fold cross-validation design. The table shows the classification accuracy for iRRD-GA, with the number of selected features in brackets, for each classifier.

Table 4.9: Comparison of the current State-of-the-art results and iRRD-GA using 5-fold cross-validation (best results shown in bold).

	Related Work				iRRD-GA	
Dataset	Reference	#Features	Accuracy	Year	#Features	Accuracy
Bone	[GZL+16]	7	85	2016	5	**91.92**
Brain	[BCSMAB+14c]	10	91	2014	2.8	**100**
Breast	[GBS+19]	50	88.82	2018	5.6	**91.58**
CNS	[XH19]	9	92.34	2019	3.8	**98.33**
Colon	[MNSMN17]	9	99.81	2017	2.2	**100**
Leukemia1	[BCSMAB+14c]	50	92	2014	1	**100**
Leukemia2	[MNSMN17]	2	100	2017	1.2	**100**
Leukemia5	[GBS+19]	4	98.67	2019	1	**100**
Leukemia6	[SIM11]	4	100	2011	2.8	**100**
Lung3	[BCSMAB+14c]	50	72	2014	10.4	**90.92**
Lung4	[XH19]	12	97.45	2019	7.4	**99.52**
Lymphoma1	[BCSMAB+14c]	10	98	2014	1.8	**100**
Lymphoma2	[GBS+19]	3	98.75	2019	2.4	**100**
Lymphoma3	[XH19]	9	92.23	2019	2	**100**
Ovarian	[BCSMAB+14c]	31	**100**	2014	2.2	99.61
Prostate2	[GZL+16]	3	93	2016	2.2	**98.05**
Prostate4	[GBS+19]	3	98.10	2019	2.4	**100**
SRBCT	[GBS+19]	5	100	2019	3.4	**100**

setups, as multi-layer perceptron classifier outperforms logistic regression by 2.5%, putting the latter into second place. Naive Bayes and KNN, once again struggle to compete with the other classifiers. When using the features selected by Table 4.15, unlike in other setups, the subset boosts the accuracy in 6 classifiers, was roughly equal accuracy on SVM, but logistic regression took a loss in prediction of 3.5%. The breast, CNS and Lung3 datasets seem to be the most challenging for most of the classifiers, except multi-layer perceptron, which could indicate that using a rectified linear unit as the activation function helped separating the classing in a non-linear manner. Looking at the iRRD-GA results (see Table 4.16), it can be seen that 6 out of the 8 classifiers scored a perfect accuracy on all 24 datasets. Naive Bayes failed to get a perfect score on 4 of the datasets, while logistic regression missed it on 2 of the datasets. Most of the classifiers only required on average 1 feature to successfully classify all the test samples, while Naive Bayes needed 1.5. The reason for the low requirement of features is possible due to using all but one sample as the training dataset. When we compare iRRD-GA with the current state-of-the-art (see Table 4.17),

Table 4.10: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 10-fold cross-validation design. These are the baseline results using no feature selection, hence all features are selected.

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	86.18	76.93	73.89	74.44	80.26	79.8	83.24	79.22	82.16	83.27
Brain	91.81	86.11	80.14	86.39	82.78	82.92	91.81	93.06	91.81	88.47
Breast	82.0	68.44	69.56	55.33	56.11	56.22	67.33	69.67	79.11	74.78
CNS	80.0	71.67	61.67	56.67	61.67	68.33	71.67	68.33	81.67	81.67
Colon	90.24	77.14	85.71	77.38	74.76	69.76	82.14	91.9	88.57	91.9
Leukemia1	100.0	100.0	94.49	93.65	96.03	93.65	100.0	97.56	100.0	99.17
Leukemia2	98.57	91.96	85.0	88.93	87.5	84.64	98.57	97.14	94.64	98.57
Leukemia3	96.07	95.89	88.87	85.18	84.76	83.33	97.5	89.29	96.07	97.32
Leukemia4	94.23	90.24	84.35	84.17	80.3	78.63	91.55	88.87	91.9	93.15
Leukemia5	98.57	93.04	98.57	86.07	84.64	84.64	98.57	98.57	94.82	95.89
Leukemia6	98.75	94.23	93.15	84.46	82.8	90.06	95.83	94.17	97.08	97.5
Lung1	100.0	100.0	100.0	98.89	98.89	98.89	100.0	100.0	100.0	100.0
Lung2	98.89	98.89	99.44	96.14	93.36	93.92	99.44	100.0	99.44	98.33
Lung3	82.22	68.48	65.2	61.93	65.12	66.23	76.46	73.68	77.95	77.4
Lung4	95.59	95.21	93.66	93.14	92.76	91.21	93.69	94.14	95.19	88.67
Lymphoma1	98.0	85.5	72.5	69.5	80.0	67.5	94.0	96.0	92.0	88.0
Lymphoma2	97.32	89.46	80.36	80.89	89.64	86.96	97.5	90.89	93.39	100.0
Lymphoma3	97.5	90.89	80.36	84.11	88.21	88.21	97.5	98.75	97.32	98.75
Ovarian	100.0	99.6	87.82	94.12	93.71	93.72	100.0	97.66	97.63	99.23
Prostate1	95.18	87.36	61.73	86.36	84.45	84.36	92.18	92.18	95.09	95.18
Prostate2	96.0	89.18	60.36	75.45	83.36	81.18	90.0	80.09	93.0	91.09
Prostate3	94.12	85.27	55.27	81.65	84.84	83.24	91.92	82.36	94.07	95.6
Prostate4	95.09	80.18	61.64	80.27	81.36	85.27	93.09	84.27	94.09	92.09
SRBCT	100.0	86.94	100.0	92.78	93.89	94.03	100.0	100.0	98.75	95.28
Average	94.43	87.61	80.57	82.0	83.38	82.78	91.83	89.91	92.74	92.55

we see that our proposed method outperforms the current results. We note that 8 out of the 17 related works achieved the same 100% accuracy as we did and another 8 was within 2% of it, with only [SAEF17] scoring outside this on the CNS dataset with 86.67%. Where we really excelled is in the number of informative features retained to achieve our results. A full table of the related work results using leave-one-out cross-validation can be found in the appendix (Tables A.7 and A.8).

4.6.1 Other Experiments

When we were deciding the final setup for iRRD-GA, we carried out a range of experiments on 2 microarray datasets, Colon and Leukemia2. The first set

Table 4.11: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 10-fold cross-validation design. The classification results are from the features selected, from step 1 of the algorithm, iRRD, and the number of features selected are given in the table.

Dataset	#Features	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	25.4	78.63	76.31	75.69	73.37	74.48	77.42	77.39	82.65	81.54	75.1
Brain	27.7	90.69	77.92	83.47	83.61	84.58	87.08	89.58	95.28	87.22	85.0
Breast	28	75.67	66.33	65.44	73.67	69.44	70.44	73.44	87.56	72.67	71.56
CNS	23.9	75.0	63.33	58.33	56.67	58.33	65.0	71.67	80.0	71.67	73.33
Colon	19.4	85.48	82.14	87.14	77.38	80.95	82.86	87.14	90.24	83.81	84.05
Leukemia1	25.7	100.0	99.23	100.0	100.0	100.0	100.0	99.23	100.0	99.23	98.4
Leukemia2	24	95.89	89.11	91.79	91.79	91.79	91.79	97.32	100.0	91.96	97.5
Leukemia3	25.9	97.32	93.39	96.07	95.89	94.82	92.32	98.57	97.32	96.07	98.57
Leukemia4	29.4	91.96	90.48	91.96	84.4	84.76	84.76	89.94	92.98	93.15	87.62
Leukemia5	23.2	95.89	93.39	93.21	93.04	93.04	93.04	95.89	97.14	94.82	98.75
Leukemia6	27.4	96.07	90.65	93.15	92.14	91.73	88.39	90.42	100.0	91.73	95.83
Lung1	23.6	100.0	100.0	96.89	100.0	99.0	99.0	100.0	100.0	100.0	100.0
Lung2	26.1	98.33	98.33	98.33	96.67	96.7	96.7	98.33	99.44	98.89	98.33
Lung3	27.3	72.05	69.44	65.12	62.95	64.04	64.09	71.52	79.04	74.82	74.3
Lung4	24.7	92.59	90.1	91.3	91.78	92.18	91.66	94.11	89.25	93.69	85.19
Lymphoma1	26.4	92.0	81.0	86.0	79.5	82.0	86.0	92.0	96.0	86.0	84.0
Lymphoma2	23.8	91.96	86.79	86.79	89.64	86.96	84.11	91.96	96.07	89.29	89.64
Lymphoma3	24	86.79	82.86	83.04	88.21	88.39	88.21	90.71	93.39	85.54	89.46
Ovarian	27.2	99.6	98.42	95.68	98.8	98.02	98.02	99.6	99.2	98.42	99.22
Prostate1	25.3	94.18	91.27	86.36	87.36	88.18	90.18	93.18	94.18	94.09	92.09
Prostate2	25.2	91.09	86.09	85.09	83.45	84.36	84.45	92.09	93.0	90.09	90.09
Prostate3	25.5	91.21	87.53	59.73	84.73	86.92	89.73	94.18	95.66	92.58	91.98
Prostate4	25	93.09	94.09	84.18	84.36	86.18	89.18	92.09	95.09	96.09	92.18
SRBCT	24.1	97.5	83.19	97.5	97.5	96.25	96.25	97.5	100.0	94.03	95.14
Average	25.34	90.96	86.31	85.51	86.12	86.38	87.11	90.74	93.9	89.89	89.47

of experiments was to determine a good parameter for the dominance ratio, between 2 possibilities 0.5 and 1.0, where the latter meant a feature was totally dominated, while the former meant it was only half dominated. A feature being totally dominated was the obvious choice, but we wanted to explore if it was only half dominated, would it add a little noise to the subset for the genetic algorithm helping to select a more robust final subset for the classifier. The bar-charts are located in the appendix section. When we compare barcharts B.1 and B.2, B.3 and B.4, B.5 and B.6 on the colon dataset, and barcharts B.7 and B.8, B.9 and B.10, B.11 and B.12 on the leukemia2 dataset, it can be seen that adding extra noise using only half domination has a decrease in overall accuracy across the majority of classifiers. For this reason, we decide to select 1.0 as the dominance ratio, but believe having a parameter that could be tuned depending on the data is a valuable part of the algorithm for future use.

Dataset	LR	DT	NB	KNN-1	KNN-3	KNN-5	SVM	MLP	RF	ADA
Bone	90.78 (2.9)	95.95 (3.3)	88.99 (3.9)	98.86 (4.7)	98.82 (5.5)	95.95 (4.2)	93.66 (3.4)	93.63 (4.4)	98.86 (3.9)	97.71 (4)
Brain	98.89 (2.2)	100.0 (1.8)	97.64 (2.8)	100.0 (1.8)	100.0 (1.7)	100.0 (1.8)	100.0 (1.9)	100.0 (1.6)	100.0 (1.9)	100.0 (2.1)
Breast	95.89 (3.8)	96.89 (3.2)	91.78 (4.1)	100.0 (2.3)	100.0 (2.8)	100.0 (4.3)	99.0 (3.9)	100.0 (3.3)	96.89 (3.1)	96.89 (4)
CNS	95.0 (2.6)	100.0 (2.1)	91.67 (5.2)	100.0 (1.5)	100.0 (1.8)	100.0 (1.8)	100.0 (2.2)	100.0 (2.1)	98.33 (1.7)	98.33 (2.4)
Colon	100.0 (1.9)	100.0 (1.7)	96.67 (2.4)	100.0 (1.2)	100.0 (1.3)	100.0 (1.1)	100.0 (1.5)	100.0 (1.5)	100.0 (1.3)	100.0 (1.3)
Leukemia1	100.0 (1)	100.0 (1)	100.0 (2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia2	100.0 (1.2)	100.0 (1.5)	100.0 (2.2)	100.0 (1.1)	100.0 (1.1)	100.0 (1.2)	100.0 (1.1)	100.0 (1.1)	100.0 (1.2)	100.0 (1.3)
Leukemia3	100.0 (2)	100.0 (3)	100.0 (2.7)	100.0 (2)	100.0 (2)	100.0 (2)	100.0 (2.2)	100.0 (2.2)	100.0 (2.1)	100.0 (2)
Leukemia4	100.0 (2.3)	100.0 (2.3)	100.0 (3.1)	100.0 (1.9)	100.0 (2)	100.0 (2)	100.0 (2.1)	100.0 (2.3)	100.0 (2.3)	100.0 (2.4)
Leukemia5	100.0 (1.1)	100.0 (1.1)	100.0 (2.1)	100.0 (1.1)	100.0 (1)	100.0 (1.1)	100.0 (1)	100.0 (1)	100.0 (1.1)	100.0 (1)
Leukemia6	100.0 (2)	100.0 (1.9)	100.0 (2.8)	100.0 (1.6)	100.0 (1.9)	100.0 (2)	100.0 (1.8)	100.0 (1.6)	100.0 (1.9)	100.0 (2.1)
Lung1	100.0 (1)	100.0 (1)	100.0 (2.1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung2	100.0 (1.4)	100.0 (1.3)	99.44 (2.3)	100.0 (1.2)	100.0 (1.3)	100.0 (1.4)	100.0 (1.2)	100.0 (1.1)	100.0 (1.2)	100.0 (1.2)
Lung3	91.96 (5.5)	94.62 (4.5)	86.05 (3.3)	94.62 (3.2)	95.67 (3.4)	95.12 (4.2)	92.51 (3.1)	94.12 (6.1)	96.26 (5.1)	95.64 (4.2)
Lung4	99.52 (5.7)	99.52 (3.8)	98.57 (7.9)	99.05 (3.7)	100.0 (4.8)	99.05 (4.4)	99.52 (4.9)	97.09 (7)	99.52 (3.7)	97.09 (5.3)
Lymphoma1	100.0 (1.4)	100.0 (1.5)	100.0 (2)	100.0 (1.1)	100.0 (1.1)	100.0 (1.1)	100.0 (1.4)	100.0 (1.1)	100.0 (1.3)	100.0 (1.2)
Lymphoma2	100.0 (2.7)	100.0 (2)	98.75 (2.5)	100.0 (1.4)	100.0 (1.6)	100.0 (1.7)	100.0 (1.9)	100.0 (1.7)	100.0 (2)	100.0 (2)
Lymphoma3	100.0 (1.7)	100.0 (1.9)	98.57 (2.6)	100.0 (1.5)	100.0 (1.7)	100.0 (1.7)	100.0 (1.7)	100.0 (1.6)	98.75 (1.5)	100.0 (1.6)
Ovarian	100.0 (1.8)	100.0 (1.7)	100.0 (2.4)	100.0 (1.6)	100.0 (1.9)	100.0 (1.9)	100.0 (1.6)	100.0 (1.6)	100.0 (1.7)	100.0 (1.8)
Prostate1	99.0 (1.9)	100.0 (2.2)	98.09 (2)	100.0 (1.5)	100.0 (1.6)	100.0 (1.9)	100.0 (1.8)	100.0 (1.7)	99.0 (2)	100.0 (2)
Prostate2	100.0 (1.8)	100.0 (1.9)	99.0 (2.1)	100.0 (1.5)	100.0 (1.5)	100.0 (1.6)	100.0 (1.5)	100.0 (1.5)	100.0 (1.5)	100.0 (2.2)
Prostate3	98.57 (2.2)	99.29 (2.5)	96.37 (2.8)	100.0 (2.2)	99.29 (2.5)	99.29 (2.5)	99.29 (2.7)	99.29 (2.6)	99.29 (2.7)	99.29 (2.6)
Prostate4	98.0 (1.7)	100.0 (2.2)	98.0 (2.1)	100.0 (1.6)	100.0 (1.4)	100.0 (1.9)	100.0 (2.2)	99.0 (1.4)	100.0 (1.8)	100.0 (2.1)
SRBCT	100.0 (2.4)	100.0 (2.4)	100.0 (3.2)	100.0 (1.9)	100.0 (2.3)	100.0 (2.4)	100.0 (2.5)	100.0 (2)	100.0 (2.4)	100.0 (2.7)
Average	98.65 (2.26)	99.43 (2.16)	97.48 (2.94)	99.69 (1.82)	99.74 (2.01)	99.56 (2.09)	99.33 (2.07)	99.3 (2.19)	99.45 (2.06)	99.37 (2.23)

Table 4.12: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under 10-fold cross-validation design. The table shows the classification accuracy for iRRD-GA, with the number of selected features in brackets, for each classifier.

Table 4.13: Comparison of the current State-of-the-art results and iRRD-GA using 10-fold cross-validation (best results shown in bold).

	Related Work				iRRD-GA	
Dataset	Reference	#Features	Accuracy	Year	#Features	Accuracy
Bone	[WMF+19]	38.8	83.71	2019	4.7	**98.86**
Brain	[BSZ17]	50	84.7	2017	1.6	**100**
Breast	[DPHC18]	4.2	92.89	2018	2.3	**100**
CNS	[BHDH+11]	3	99.3	2011	1.5	**100**
Colon	[TJGNA08]	2	100	2008	1.1	**100**
Leukemia1	[WMF+19]	12.92	100	2019	1	**100**
Leukemia2	[DR03]	4	100	2003	1.1	**100**
Leukemia3	[KAAAJ11]	10	100	2011	2	**100**
Leukemia4	[SMM16]	13.08	86.38	2016	1.9	**100**
Leukemia6	[SMM16]	14.9	99.30	2016	1.6	**100**
Lung1	[Das17]	5	100	2017	1	**100**
Lung2	[DPHC18]	4.1	100	2018	1.1	**100**
Lung3	[BSZ17]	50	73.5	2017	5.1	**96.26**
Lung4	[MM16]	10	100	2016	4.8	**100**
Lymphoma1	[ALA16]	2	100	2016	1.1	**100**
Lymphoma2	[GBGK12]	30	95	2012	1.4	**100**
Lymphoma3	[MM16]	10	100	2016	1.5	**100**
Ovarian	[ALA16]	2	100	2016	1.6	**100**
Prostate1	[KAAAJ11]	45	99.6	2011	1.5	**100**
Prostate2	[DPHC18]	2	100	2018	1.5	**100**
Prostate3	[SMM16]	14.5	99.85	2016	2.2	**100**
Prostate4	[MM13]	3	96.08	2013	1.4	**100**
SRBCT	[MM16]	5	100	2016	1.9	**100**

Next we look at the reasoning behind the statistic selection, with our theory being having 2 very different and distinct statistics will allow the algorithm to retain features from very different perspectives. Comparing the 3 types of statistical designs on both datasets; Cramer's ϕ (see Figures B.2 and B.8), Mutual Information (see Figures B.4 and B.10) and using both statistics (see Figures B.6 and B.12), we see that cramer's ϕ scores the lowest accuracy, followed by mutual information just beaten on the dual statistic setup. Using both statistics, based on different fundamentals, information theory and $\tilde{\chi}^2$, helps to achieve an overall better classification accuracy averaged across the contrasting classifiers.

The final experiment was to evaluate whether it was better to use the same

Table 4.14: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under leave-one-out cross-validation design. These are the baseline results using no feature selection, hence all featreus are selected.

Dataset	LR	DT	Naive	KNN-1	KNN-3	KNN-5	SVM	MLP
Bone	87.28	83.82	73.41	75.72	79.77	78.61	83.82	79.19
Brain	92.94	88.24	81.18	89.41	84.71	82.35	91.76	97.65
Breast	81.05	86.32	68.42	55.79	55.79	55.79	69.47	100.0
CNS	78.33	83.33	60.0	58.33	55.0	61.67	78.33	75.0
Colon	90.32	75.81	85.48	77.42	74.19	72.58	79.03	98.39
Leukemia1	100.0	100.0	93.75	93.75	95.31	95.31	100.0	100.0
Leukemia2	98.61	90.28	87.5	88.89	87.5	84.72	98.61	100.0
Leukemia3	98.61	94.44	84.72	86.11	84.72	81.94	97.22	98.61
Leukemia4	95.83	93.06	83.33	84.72	80.56	79.17	93.06	98.61
Leukemia5	98.61	93.06	98.61	84.72	86.11	83.33	98.61	100.0
Leukemia6	98.61	97.22	94.44	86.11	81.94	87.5	97.22	100.0
Lung1	100.0	100.0	100.0	98.96	98.96	98.96	100.0	100.0
Lung2	99.45	97.79	99.45	95.58	93.92	93.92	99.45	100.0
Lung3	85.56	78.61	64.17	63.1	68.45	68.98	74.33	94.65
Lung4	96.06	97.04	93.6	92.61	93.1	91.13	93.6	99.01
Lymphoma1	97.87	89.36	68.09	65.96	68.09	68.09	89.36	100.0
Lymphoma2	97.4	84.42	79.22	81.82	88.31	88.31	97.4	98.7
Lymphoma3	97.4	87.01	80.52	84.42	88.31	90.91	97.4	100.0
Ovarian	100.0	99.6	88.14	94.86	94.47	93.68	100.0	100.0
Prostate1	95.1	81.37	61.76	85.29	83.33	84.31	92.16	99.02
Prostate2	96.08	88.24	60.78	73.53	80.39	82.35	91.18	95.1
Prostate3	94.12	89.71	55.15	77.94	84.56	81.62	92.65	100.0
Prostate4	95.1	83.33	61.76	84.31	83.33	83.33	90.2	99.02
SRBCT	100.0	85.54	98.8	92.77	93.98	92.77	100.0	100.0
Average	94.76	89.48	80.1	82.17	82.7	82.56	91.87	97.21

classifier for (i) chromosomes evaluation in the genetic algorithm, and (ii) predicting the final model output. The other option was to use a different classifier for each part of the model construction, output prediction and genetic algorithm evaluator. As we can see from Figures B.6 and B.12, in most cases using the same statistics for both parts of the algorithm had a positive improvement in the classification prediction, although in some cases, the results were very close. This showed that when iRRD-GA selected a *good* subset of features for the final prediction model, the choice of classifier had less of an impact on the resulting accuracy.

Table 4.15: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under leave-one-out cross-validation design. The classification results are from the features selected, from step 1 of the algorithm, iRRD, and the number of features selected are given in the table.

Dataset	#Features	LR	DT	Naive	KNN-1	KNN-3	KNN-5	SVM	MLP
Bone	25	82.66	85.55	76.88	78.03	76.88	77.46	83.82	92.49
Brain	27.5	89.41	88.24	80.0	84.71	85.88	87.06	90.59	98.82
Breast	28	68.42	83.16	64.21	55.79	57.89	55.79	72.63	100.0
CNS	23.9	73.33	75.0	46.67	48.33	53.33	51.67	80.0	95.0
Colon	19.5	91.94	87.1	85.48	74.19	80.65	80.65	85.48	98.39
Leukemia1	25.9	100.0	99.22	100.0	100.0	100.0	100.0	100.0	100.0
Leukemia2	23.4	97.22	91.67	94.44	97.22	94.44	93.06	97.22	100.0
Leukemia3	26.2	97.22	94.44	95.83	94.44	95.83	94.44	95.83	100.0
Leukemia4	28.4	91.67	88.89	93.06	93.06	93.06	90.28	91.67	98.61
Leukemia5	22.8	94.44	90.28	94.44	90.28	87.5	88.89	95.83	100.0
Leukemia6	27.6	94.44	87.5	94.44	91.67	90.28	90.28	94.44	100.0
Lung1	23.9	97.92	100.0	98.96	98.96	98.96	100.0	98.96	100.0
Lung2	26.7	98.34	98.9	98.9	98.34	97.79	97.79	98.34	98.34
Lung3	26.9	70.05	82.35	62.57	57.75	61.5	62.57	74.87	99.47
Lung4	25	93.6	96.06	92.12	91.13	93.1	93.1	94.58	95.57
Lymphoma1	26.3	89.36	87.23	72.34	82.98	80.85	78.72	93.62	100.0
Lymphoma2	23.8	92.21	88.31	87.01	85.71	88.31	84.42	90.91	100.0
Lymphoma3	23.8	94.81	90.91	89.61	90.91	88.31	89.61	94.81	100.0
Ovarian	27.8	98.81	97.23	97.23	100.0	99.21	99.21	99.21	99.21
Prostate1	24.6	92.16	89.22	88.24	82.35	84.31	88.24	94.12	99.02
Prostate2	24.8	93.14	89.22	88.24	76.47	85.29	87.25	93.14	100.0
Prostate3	25.6	94.12	92.65	60.29	88.24	85.29	85.29	95.59	100.0
Prostate4	24.5	93.14	91.18	88.24	84.31	88.24	88.24	92.16	98.04
SRBCT	24.5	98.8	90.36	98.8	97.59	97.59	97.59	95.18	100.0
Average	25.27	91.13	90.19	85.33	85.1	86.02	85.9	91.79	98.87

These set of experiments helped us to determine that for microarray data: a dominance ratio of 1.0 was best, using a dual statistic design gave us a robust subset of features resulting in an overall better classification model, and finally using the same classifier throughout the algorithm was deemed most suitable.

4.7 Conclusion

In the quest of improving cancer detection using microarray data, we have shown our proposed approach can achieve very high classification accuracy, which could be used as a tool to assist medical experts in the fight against can-

Dataset	LR	DT	Naive	KNN-1	KNN-3	KNN-5	SVM	MLP
Bone	95.95 (1.3)	100.0 (1.1)	94.8 (1.2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1.1)	100.0 (1)
Brain	100.0 (1)	100.0 (1)	100.0 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Breast	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
CNS	96.67 (1.3)	100.0 (1)	93.33 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1.1)	100.0 (1)
Colon	100.0 (1)	100.0 (1)	100.0 (1.4)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia1	100.0 (1)	100.0 (1)	100.0 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia2	100.0 (1)	100.0 (1)	100.0 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia3	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia4	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia5	100.0 (1)	100.0 (1)	100.0 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Leukemia6	100.0 (1)	100.0 (1)	100.0 (1.6)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung1	100.0 (1)	100.0 (1)	100.0 (1.1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung2	100.0 (1)	100.0 (1)	100.0 (1.2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung3	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lung4	100.0 (1.1)	100.0 (1)	99.51 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1.1)	100.0 (1)
Lymphoma1	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lymphoma2	100.0 (1)	100.0 (1)	98.7 (1.3)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Lymphoma3	100.0 (1)	100.0 (1)	100.0 (1.2)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Ovarian	100.0 (1)	100.0 (1)	100.0 (1.4)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Prostate1	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Prostate2	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Prostate3	100.0 (1)	100.0 (1)	100.0 (1.4)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Prostate4	100.0 (1)	100.0 (1)	100.0 (1.5)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
SRBCT	100.0 (1)	100.0 (1)	100.0 (1.7)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)	100.0 (1)
Average	99.69 (1.03)	100.0 (1.0)	99.43 (1.4)	100.0 (1.0)	100.0 (1.0)	100.0 (1.0)	100.0 (1.01)	100.0 (1.0)

Table 4.16: The accuracy results of the 24 microarray dataset, using 11 classifiers, and performed under leave-one-out cross-validation design. The table shows the classification accuracy for iRRD-GA, with the number of selected features in brackets, for each classifier.

Table 4.17: Comparison of the current State-of-the-art results and iRRD-GA using leave-one-out cross-validation (best results shown in bold).

	Related Work				iRRD-GA	
Dataset	Reference	#Features	Accuracy	Year	#Features	Accuracy
Breast	[TES17]	5127	100	2017	1	**100**
CNS	[SAEF17]	38	86.67	2017	1	**100**
Colon	[ABA15b]	10	98.38	2015	1	**100**
Leukemia2	[ABA15b]	4	100	2015	1	**100**
Leukemia3	[ABA15b]	8	100	2015	1	**100**
Leukemia5	[GBS+19]	4	98.61	2019	1	**100**
Leukemia6	[MODY11]	4	100	2011	1	**100**
Lung1	[ABA15b]	4	100	2015	1	**100**
Lung2	[LJL+14]	30	99.45	2014	1	**100**
Lymphoma1	[DB17]	21	99.70	2017	1	**100**
Lymphoma2	[GBS+19]	3	98.70	2019	1	**100**
Lymphoma3	[LH08]	6	100	2008	1	**100**
Prostate1	[CYWY11]	24.7	99.22	2011	1	**100**
Prostate2	[LH08]	6	98.04	2010	1	**100**
Prostate3	[SAEF17]	26	100	2017	1	**100**
Prostate4	[GBS+19]	3	99.02	2019	1	**100**
SRBCT	[LL11]	6	100	2011	1	**100**

cer. Furthermore, in bioinformatics, any improvements in the research area of microarray data classification is very important and can have huge extended benefits in a multitude of domains. In this chapter, we proposed a novel hybrid two-step method, improved Relevance-Redundancy Dominance with a Genetic Algorithm (iRRD-GA) for feature selection. Step one, iRRD, efficiently and significantly reduces the original dataset to a small set of relevant genes using Mutual Information and Cramer's ϕ as the statistics to filter the original high dimensional dataset. While step two, GA, uses a stochastic genetic algorithm to find a *good* minimum set of important features.

We thoroughly checked the robustness of the proposed approach by analysing it on a range of microarray datasets, with various class sizes and dimensionality. IRRD-GA was evaluated using a multiple number of classifiers under an array of different experimental cross-validation design to compare with the current state-of-the-art in the literature. The proposed method consistently obtained new state-of-the-art classification accuracy results while retaining a minimum subset of informative genes. In future work we would like to explore the par-

allelization of the method as we believe iRRD-GA could be applied to NP-Hard problems in other areas, including "Big-Data" problems.

In the next chapter, Chapter 5, we carry on the theme of reducing classification problem complexity in order to help with model efficiency, by implementing a pruning approach to reduce the dimensionality of convolutional neural networks (CNN). But, due to using the target classes to help remove redundant and irrelevant parts of the classification task being too computational costly, we need to look at a different approach. We introduce our novel pruning algorithm, which iteratively prunes irrelevant parts of the CNN.

Chapter 5

Pulse-Net: Dynamic Compression of Convolutional Neural Networks

5.1 Motivation

After showing the importance of reducing the complexity of problems, and helping improve the efficiency of classifiers in the previous 3 chapters, we now look at ways to carry on the work but within the area of deep learning. Various attempts were made to incorporate the idea of dominance when trying to prune neural networks, but due to the need for using the dataset targets as part of the RRD type algorithms (see Sections 2.5.2, 3.5.2 and 4.5.2), it made the idea too computationally expensive to implement. Therefore, a new novel idea was needed to further the theme of the thesis, reducing parameters resulting in better models, in the scope of deep learning classification research.

In this chapter, we look at reducing the model complexity of deep learning methods. Convolutional Neural Networks (CNNs) are used in a range of computer vision tasks, with state-of-the-art CNNs such as AlexNet and VGG16 constructed using a large number of parameter and multiply-add operations (MACs). These tasks require high computational power and high energy requirements to run the CNNs, making them unsuitable for deployment on Internet of Things devices. To overcome this issue and facilitate the use of CNNs on these resource-constrained devices, compression technology through pruning research has gained momentum and is an important tool for improving performance during inference. Our work focuses on pruning unwanted filters and nodes in all layers of a network. The network is pruned iteratively during training via a novel ap-

proach we call Pulse-Net, and a significant number of filters/nodes are removed while ensuring any loss in accuracy is within a predetermined range. The unpruned network can be extracted from the original structure for the inference stage. This novel method has an easy-to-set parameter to control the trade-off between accuracy and compression. Pulse-Net gives greater compression, while maintaining competitive accuracy loss, than other reported methods like, efficient convnets, ThiNet and Cross-Entropy Pruning. It also has better robustness against adversarial attacks than other compression and pruning techniques.

5.2 Introduction

Recent years have seen the explosion of increasingly deep neural networks, which achieve start-of-the-art results in areas including computer vision. The rapid growth of deep convolutional neural networks (CNNs) is due to hardware developments in the form of powerful GPUs, software developments in the form of stochastic gradient descent and new network architectures, and the public availability of large labelled datasets such as the ImageNet classification tasks.

However, because deep CNNs rely heavily on powerful GPUs and consume a great deal of memory, their practical uses may be limited, although there have been some advances in overcoming this, such as the Intel-Movidius Neural Compute Stick, they are not without their limitiations. AlexNet [KSH12] has about 61 million parameters and needs over 200 MB of storage, while VGG16 [SZ14] has 138 million parameters requiring 500 MB. The more parameters the model has, the more memory it consumes and the more energy is needed during inference. This is particularly important when these networks are deployed on mobile devices, and memory consumption is the key resource for usage on the cloud. Cost and power requirements can also be important determining factors when considering introducing CNNs for Internet of Things (IoT) applications such as health monitoring, indoor localization and autonomous driving. Inference time can be just as important as accuracy for online image recognition where thousands of images per second may require analysis. We show that by compressing/pruning the network we can greatly reduce the number of parameters, leading to a significant reduction in the number of floating point operations per second (FLOPs), which is directly associated to inference time.

As deep neural networks become deeper and wider, methods to prune filters/nodes and compress the network structure have gained interest. Most recent work in

CNN compression is focused on reducing the number and size of weights or parameters, but does not take into account the value of an entire filter/node (learnable weights of the network). According to Denil *et al.* [DSD+13] and Hinton *et al.* [HSK+12], deep neural networks are known to be overparameterized, which facilitates convergence to good local minima of the loss function during training. After training, these redundant parameters can be removed with little loss in accuracy. There are two general approaches to compressing a network: during training [CBD15, CHS+16, IHM+16] or after training [CWT+15, GYC16, HPTD15, HMD15, WWW+16]. Our proposal, called Pulse-Net, falls in the first group.

We demonstrate the robustness, efficiency and strength of our proposed approach on a range of computer vision tasks of varying degrees of difficulty (CIFAR-10, CIFAR100, Tiny-ImageNet), using well-known CNN structures of different depths and widths. We compress and evaluate these structures, showing how Pulse-Net can significantly reduce the model size, runtime and energy consumption, while approximately maintaining accuracy within a predefined threshold. The advantages of these compressed models are that they are easier to run on embedded IoT devices, need less energy for computation and use less bandwidth for updating. Adversarial images are tested on both the original and the compressed networks, demonstrating the effects of adversarial attacks on pruned networks.

A number of approaches have been suggested to help reduce the computational overheads of CNNs, including sparsity [WWW+16, LWF+15], weight binarization [CHS+16, RORF16] and network quantization [HMD15, TM18]. Although these methods have achieved good results they usually need additional software and/or hardware, thus increasing problem complexity and making them impractical for real-world applications. [RBK+14, HVD15] presented a student-teacher approach called FitNets, based on the idea of knowledge distillation. Network pruning has shown significant promise in this research area, which includes the pruning of weights [HPTD15, LAT18, GYC16], filters [HKD+18, HZYN18, LKD+16] and channels [LWL17, HZS17, LLS+17]. It has the added bonus of being implementable on current software without requiring additional support. Specially designed networks can also improve CNN efficiency [IHM+16, HZC+17].

Channel pruning is a form of structured pruning, and is a popular approach to CNN compression [CTSO18, AAY17]. It follows a standard method of remov-

ing channel links then fine-tuning the network, repeating until some decision boundary is met. At each iteration a smaller network is created that maintains accuracy with negligible loss. This inspires an interesting question: is it necessary to prune a large network to reach high accuracy, or can we simply start with a small network and achieve the same result with faster training? Current research yields contradictory evidence. [FC18] found that sub-networks (usually found through pruning, and contained in dense, randomly-initialized, feed-forward networks) when trained in isolation can reach accuracy similar to the original network. On the other hand [LSZ+18] found, using the same experimental design as [FC18], that random initialization of the pruned network was sufficient to achieve a similar performance, with the only requirement being that the standard 0.1 learning rate was used rather than a smaller one (0.01). This meant [FC18] required the use of the original initialization of the pruned model, whereas [LSZ+18] successfully used random initialization. These results show how unpredictable pruning CNNs with or without random re-initialization can be.

The main contributions of this research are as follows. Firstly, we introduce a novel approach called Pulse-Net for pruning filters and nodes during training. Secondly, we demonstrate that this achieves close to state-of-the-art classification accuracy on the well-known datasets CIFAR-10, CIFAR100 and Tiny-ImageNet while using only a fraction of the parameters. Thirdly, we show how the compressed network can be extracted and loaded onto a new narrower CNN for the inference phase, and that simulations of its usage on an IoT device shows substantial improvements in computational speed and energy efficiency. Finally, robustness under adversarial attack of the compressed network created by Pulse-Net will be demonstrated, showing the compressed network to be just as accurate as the original network, and in some cases to have better accuracy.

This rest of this chapter is organised as follows. Section 5.3 discusses the related work on filter pruning and network compression. We explain our proposed method Pulse-Net in Section 5.4, with the descriptions of the datasets and networks used in Section 5.5, along with the experimental design. The results are given in Section 5.6, as well as their evaluation and discussions. We conclude the chapter in Section 5.7.

5.3 Related Work

Research on the compression of deep neural networks has gained pace in the last couple of years, to enable these state-of-the-art networks to run on Internet of Things devices, and to use less bandwidth for updating. As already stated, CNNs are generally over-parameterized, so pruning them helps to improve generalization, as well as achieving compression and decreasing energy consumption with little loss in accuracy. Molchanov *et al.* [MAV17], following the same idea, uses the Dropout regularization method to help decide which weights can be pruned.

Han *et al.* [HPTD15] and Guo *et al.* [GYC16] trained a network then pruned redundant weights, resulting in a sparse network. To regain accuracy this sparse network was retrained. A disadvantage of this method is that additional libraries are required to utilize sparse networks, which can cause compatibility issues and increase their computational cost. Also, though this can reduce the size of a network, it does not necessarily make the network more efficient or faster at inference time. This is due to the structure of the networks studied, AlexNet [KSH12] and VGG16 [SZ14], in which the number of parameters in the fully connected layers dominates the number of parameters in the convolutional layers. Most of the computation time during inference is spent in the convolutional layers, so removing these redundant weights results in little improvement in inference time. The recently developed Highway Networks [SIVA17] and ResNets [HZRS16] counter this by removing the fully connected layers, but this does not help with computation time which greatly increases as networks become deeper. Li *et al.* [LKD+16] proposed a filter pruning method which is more hardware-compatible: we follow this line, but we also prune nodes in the fully-connected layers to achieve greater compression.

Han *et al.* [HMD15] extended [HPTD15] by using quantization and Huffman coding in conjunction with network pruning, an approach called Deep Compression. To show its application value they released a hardware accelerator called Efficient Inference Engine where the compressed model runs more energy-efficiently [HLM+16]. Polyak and Wolf [PW15] pruned their network based on low-level channels, while Sun *et al.* [SWT16] learned a sparse network which reduced the number of parameters by 88%. The ideas of Han *et al.* [HMD15], Courbariaux *et al.* [CHS+16] and Rastegari *et al.* [RORF16] work could be applied to our compressed networks to further reduce their size.

Some recent state-of-the-art CNN architectures reduce the network parameters by reducing filter size. The VGG network [SZ14] uses 3×3 filters, while googleNet [SIVA17] and Network-in-Network [LCY13] both use 1×1 filters in some layers. Other recent research in CNN structure shows that, although adding layers increases accuracy, it is possible to skip layers in the network, resulting in further improvement in accuracy [HZRS16] [SGS15].

Hinton *et al.* [HVD15] used the outputs from a large pre-trained teacher network to train a smaller student network, with similar accuracy but faster computation. A drawback of this approach is that it requires a pre-trained network, and that further training operations are carried out on the large teacher network. Both Romero *et al.* [RBK+14] and Luo *et al.* [LZL+16] expanded on this idea, the former using not only the outputs but also representations learned by the teacher network to train a narrower and deeper student network. [LZL+16] used the weights learned by the teacher at the last hidden layer before the softmax layer, reasoning that these weights are highly correlated to the prediction classes.

Yang *et al.* [YCS17] introduced energy consumption as the metric to decide which layers to prune. This work shows that the energy a model consumes has two components: (1) energy used to carry out the MAC operations, and (2) energy needed for memory accesses. We report how Pulse-Net can effectively reduce (1) by reducing the number of MAC calculations, and reduce (2) by shrinking the network structure.

Both Wang *et al.* [WDH+18] and Ye *et al.* [YWW+18] showed that compressed CNNs can be vulnerable to adversarial attack. This is shown by testing adversarial images created using the Fast Gradient Sign Method, on both the original network and the network compressed by 60%. The accuracy of both networks were greatly reduced and the difference in accuracy between them was over 13%. This showed that their compressed network was more vulnerable to adversarial attack. We will show that Pulse-Net can reduce the network 95.63% while maintaining similar accuracy during adversarial attacks, in some cases even beating the original network.

5.4 Pulse-Net

In this section we describe Pulse-Net, our network pruning technique, and show the pseudo code behind the idea.

Pulse-Net, so-called due to the pulsing nature of the number of actively updated model parameters in time during training, is the main algorithm of our pruning method. The pulsing nature comes from the shape of both the training and validation curves, as the network is pruned and fine-tuned. The curves take a sharp dip, due to the pruning of the network, followed by a rapid recovery of accuracy, as it is fine-tuned. It prunes the network iteratively, but also allows the network to expand when the loss of accuracy is too great. To check when the network converged, a moving average with window size 10 was used to smooth out the validation loss curve. The learning rate list was $[0.1, 0.01, 0.001, 0.0001, 0.00001]$ (lr-list for short in Algorithm 6). This list was the step sizes used in the optimization process of training the network, where the learning rate started at the maximum value in the list and was stepped down through the values of the list on the condition of the validation curve converging. It should be noted that the learning rate is a configurable hyperparameter, used in the optimization part of the training process, that determines the step size at each iteration while trying to minimize the loss function. This effectively controls how quickly the network adapts to the problem. This compression and decompression of the network allows Pulse-Net to prune the network intelligently, and approach a compressed state more gently. This idea follows the use of cooling schedules in Simulated Annealing, a well-known optimization algorithm with good convergence properties. It allows Pulse-Net to explore ambitious pruning but focus on pure improvement when this fails. Another optimization algorithm that Pulse-Net takes properties from is NeuroEvolution of Augmenting Topologies (NEAT) [SM02] in which neurons can be added as necessary to regain accuracy. The full training process of Pulse-Net varies depending on the compression rate achieved, but on average takes 2–5 times longer than training a model without compression. But once the model is compressed, the inference runtime is far more efficient than an uncompressed model.

Once the network is trained, all its layers are reduced by the same percentage. This ensures that the network is pruned more quickly, by pruning both the convolutional and fully-connected layers together. The network is fine-tuned until it stabilises, normally only requiring a few loops of the training data, then the pruning and fine-tuning procedure is repeated until set break-points are met or the loss in accuracy is too great. In each layer, the filters/nodes which have the lowest average value are removed first. During the training stage, a binary mask matrix is used to simulate the removal of the corresponding filters/nodes from these redundant feature maps. Once Pulse-Net has finished the reduction

stage, the remaining relevant filters/nodes are extracted and reloaded onto a compressed network for the inference stage.

We now introduce some notation. Let α be the rate of reduction, which we initially set to 10%. This is the percentage all layers in the network will be pruned by until an accuracy limit is breached. Let λ be the maximum loss in accuracy which allows the network to continue pruning at the current α rate. This is the parameter that controls how much a network is willing to sacrifice accuracy for greater compression. For all experiments in this work we set it to 2%, which gave high compression with very little accuracy loss. Parameter β relates to the stopping criterion: we set it to 2 nodes/filters of the widest current layer of the network. This means that if α prunes less than β in this layer, Pulse-Net stops trying to further compress the network. The pseudo-code for Pulse-Net is shown in Algorithm 6.

The following bullet points give a line-by-line description of Algorithm 6:

- In line 1 we define the network loss as ρ, while testing the network on the validation set.
- In line 2 we define the network classification accuracy, on the validation set as σ.
- In line 3 we define the network current best classification accuracy stored, on the validation set as γ.
- Line 4 we initialize the reduction rate α to 10%, which is the percentage each layer of the network is reduced by during the pruning section.
- Line 5 we initialize β to 2, which is used as the pruning stopping criteria. If the widest layer has less than β filters.nodes to prune, we deem the network to be fully pruned and ends the algorithm.
- Line 6 we initialize λ to 2%, which is the maximum difference loss in classifiaction accuracy on the validation set. If the network's loss is greater than λ, then the network returns to its last best state, and is pruned at a lower rate, or if all the conditions are meet, the algorithm stops.
- In line 7 we initialize the learning rate step to 0. This is used to step through the asigned learning rates, which are used in the optimizer to train the network.
- In line 8 we initialize the learning rate list.

Algorithm 6 Pulse-Net

1: Define ρ as the network loss while using the validation set
2: Define σ as the classification accuracy of the validation set
3: Define γ as the currently stored maximum accuracy of the validation set
4: $\alpha = 10\%$
5: $\beta = 2$
6: $\lambda = 2\%$
7: Initialize lr-step $= 0$
8: Initialize lr-list = $[0.1, 0.01, 0.001, 0.0001, 0.00001]$
9: Initialize lr-rate $=$ lr-list[lr-step]
10: Train Network, using the lr-rate value, until ρ converges
11: Calculate σ and store as γ
12: **while** # Filters/Nodes of max layer removed $> \beta$ **do**
13: **while** $|\sigma - \gamma| < \lambda$ **do**
14: Remove α min[Filters] in all layers of Network
15: Reset lr-step $= 0$
16: Fine-Tune Network, using the lr-rate value, until ρ converges
17: Calculate σ
18: **if** $\sigma > \gamma$ **then**
19: $\gamma \leftarrow \sigma$
20: **else**
21: **if** lr-step $<$ length (lr-list) **then**
22: $\alpha = 0.5(\alpha)$
23: lr-step $=$ lr-step $+1$
24: lr-rate $=$ lr-list[lr-step]
25: **else**
26: **Halt**
27: **end if**
28: **end if**
29: **end while**
30: **end while**

- In line 9 we use the learning rate step to select the initial learning rate value from the list of learning rate values.
- Line 10 is where we train the network, using the learning rate for the optimizer, until the validation loss curve shows the nework has converged. This is seen when the loss of the network begins to level off, or starts to increase, while the training loss continues to decrease.
- In line 11 we calculate the network's classification accuracy, and since this is the first training of the network, we store it as the current best classification accuracy γ.
- Lines $12 - 30$ are a loop; this is exited (line 28) when the number of

filters/nodes of the widest layer is less than β.

- This loop is the main part of the algorithm, and is where the algorithm selects which filters/nodes to remove.
- Lines $13 - 29$ are a second inner loop which checks to ensure the network maintains classification acuracy within the threshold limit λ.
- In line 14 we reduce the size of each layer of the network by the current reduction percentage value α. For each layer, we first claculate the absolute value for each filter/node, and then simulate the lowest value ones (the α amount) being removed by inserting a binary matrix into the network, with zeros where the filter/node is removed and ones where the filter/nodes remains.
- Line 15 we reset the learning rate step.
- Line 16 is where the pruned network is fine-tuned to regain any loss in accuracy. The optimizer starts using the maximum learning rate, and fine-tunes the network until the validation loss curve show convergence in the network.
- Line 17 we calculate the classification accuracy on the validation set.
- Lines $18 - 28$ are an if condition that checks if the current classification accuracy on the validation set is within the threshold λ. If it is, we store it as the new best classification accuracy, otherwise we reduce the learning rate of the optimizer, and also half the percentage of the reduction rate α. This helps the algorithm approach the fully pruned state in a Simulated Annealing manner.
- Line 19 we store new best classification accuracy on validation set as the current best accuracy.
- Lines $21 - 27$ are an if condition that checks if the optimizer has used all the learning rates in the given learn rate list.
- In lines $22 - 24$ we half the reduction rate α and set the next smallest learning rate to be used in the optimizer.
- In ine 26 the algorithm stops once the the stopping criteria is statisfied.

Now that we have given a detailed description of Pulse-Net, we will give a higher level explanation to ensure that the algorithm is fully understood. We

train a network on the target dataset until the validation curve levels off. We simulated the pruning of all the layers in the network, using a set percentage, by using a binary matrix to turn on/off the lowest absolute value filters/nodes. The network is fine-tuned, using the set learning rate in the optimizer, until the validation curve levels off again. This pruning and fine-tuning is repeated, until the network is unable to recover the required drop in classification accuracy (2%). When the network is unable to recover the accuracy, we conclude it is fully pruned (compressed). At this point, the last part of the method extracts all the relevant parameters from the network to initialize the pruned network onto a smaller network for the inference phase. That is the main flow of Pulse-Net.

5.5 Methods

We now explain the experimental setup and describe the datasets and network architectures used in our experiments.

5.5.1 Experiment Design

All training and testing was carried out on the NVIDIA GeForce GTX 1080 graphics card, which has 8GB of memory. The OS used was Ubuntu 16.04.3, the Python version was 3.5.2 and the TensorFlow version was 1.4. A mini batch size of 128 was used on all experiments, and to create the validation dataset, the last 10% was used. The Stochastic Gradient Descent (SGD) optimizer was used during the training phases with a repeated step learning rate method, explained above.

To check the robustness of the networks under adversarial attack, we also test them on adversarial images constructed using the Fast Gradient Sign Method [GSS14]. Adversarial images are images with intentionally perturbed pixels with the purpose of fooling the network during inference time. We created adversarial images of CIFAR10, CIFAR100 and tiny-Imagenet using separately trained models of CifarNet, AlexNet and VGG16, dedicated to creating only these adversarial test sets. The full validation sets of these datasets were used as the adversarial test sets. To avoid confusion we name these adversarial networks Adv-CifarNet, Adv-AlexNet and Adv-VGG16. We compute statistics showing the noise required to create the adversarial images needing the most and least amount of noise for each model per dataset. These statistics include mean square error, entropy and structural similarity, which is used to compare images

for likeness. Standard statistics such as inter-quartile range, mean and standard deviation show the variation of noise required for the adversarial effect, while the range and outliers show the max and min values of noise added, along with the extreme values.

5.5.2 Benchmark Image Recognition Datasets

Table 5.1: Table comparing the 3 different image recognition data sets used in this research work.

Dataset	#Classes	#Images per class	Train/Validation/Test	Image Sizes
CIFAR10 [KH+09]	10	6000	40000 / 10000 / 10000	32×32
CIFAR100 [KH+09]	100	600	40000 / 10000 / 10000	32×32
Tiny-Imagenet [RDS+15]	200	550	90000 / 10000 / 10000	64×64

We use three benchmark image recognition datasets — CIFAR-10, CIFAR-100 and Tiny-ImageNet — to evaluate the performance of Pulse-Net, Table 5.1. Both of the CIFAR datasets are 32×32 pixel RGB images, and CIFAR-10 has 10 classes while CIFAR-100 has 100 classes. Tiny-ImageNet, a resized subset of ImageNet, contains 64×64 pixel RGB images consisting of 200 classes. Both CIFAR-10 and CIFAR-100 have 50,000 training images and 10,000 testing images, so in CIFAR-10 each class has 5000 examples, while in CIFAR-100 each class has 500 images. To create the validation sets for the CIFAR datasets, 10% of each class in the training set was randomly selected, and the remaining 90% was used as the final training set. This subset was then used as the validation set. Tiny-ImageNet was created by Stanford University as an image classification problem, run similarly to the ImageNet challenge (ILSVRC), that allows users to carry out experiments within a reasonable time. The dataset has 100,000 training images, and 10,000 testing images, giving respectively 500 and 50 samples of each class in the training and testing datasets. Similarly, to the CIFAR datasets, the Tiny-ImageNet training dataset was randomly sampled at a rate of 10% of each class, and these 10% samples were combined to create the validation dataset.

5.5.3 Convolutional Neural Networks

Figure 5.1 is an illustration of the general structure of the CNNs used in this chapter and throughout the rest of the project. As can be seen, the network takes a *car* as the input image, which is followed by a convolutional layer with an activation function and batch normalization. Convolutional layers are the

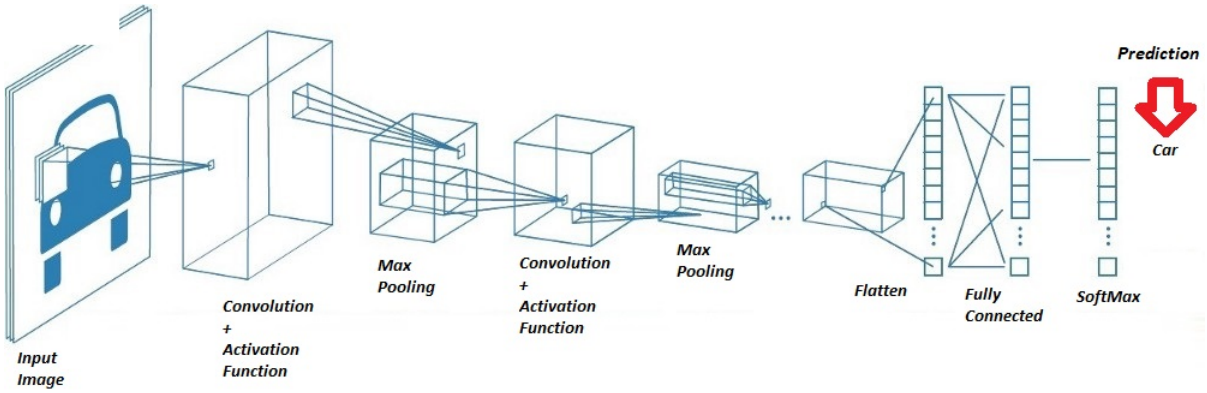

Figure 5.1: This figure shows the general architecture of the convolutional layers used throughout this thesis.

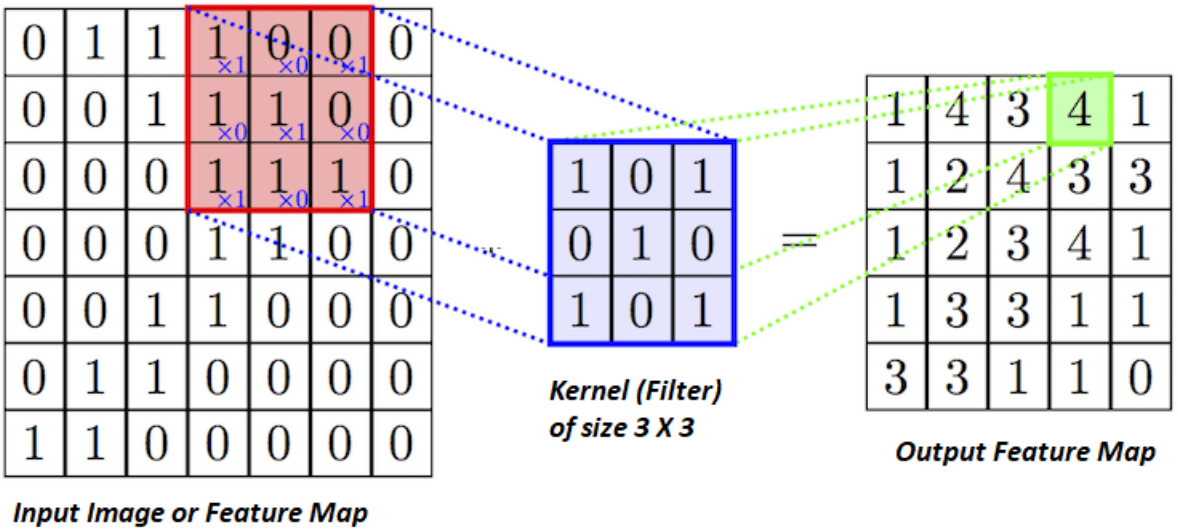

Figure 5.2: This figure shows the convolutional operation of the networks.

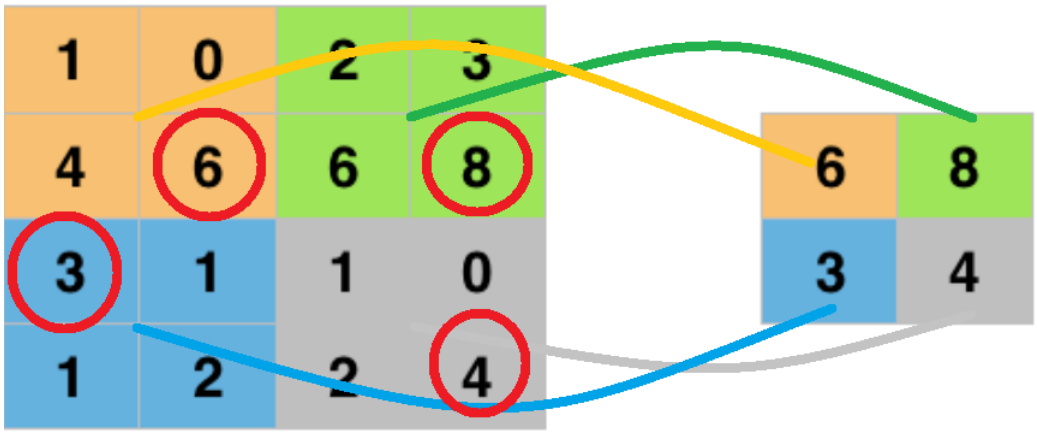

Figure 5.3: This figure shows how max pooling reduces the size of the input feature map by selecting the biggest value in a 2×2 section of the image.

backbone of CNNs, and through geometric assumptions about the data, overcome the limitation of the original dense layer networks. The convolution operation, Figure 5.2, is an element multiplication between the kernel and a same size part of the input (image or feature map). The kernel or filter is one of the trainable elements of the network, where through back propagation, the network tunes the kernel weights to achieve a better classification result. This is done by calculating the partial derivatives of the loss function with respect to the learnable weights and biases. The kernel is moved across the input image in steps called strides, in most cases a stride of 1, except for in AlexNet, where in its first convolution layer has a stride of 4. Depending on stride and filter sizes, padding of the image with zeros may be necessary, to ensure there is enough of the image to allow the filter to stride across it. Padding controls the output size of the feature maps; allowing the layer to preserve the exact spatial representational size as that of the input. After adding a trainable bias to the output feature map, it is put through batch normalization and an activation function. Batch normalization is an approach that standardizes the inputs, for each mini-batch, helping the network to converge faster and reducing the emphasis on the initialization of the weights. It also allows for a larger learning rate and acts as a regularizer. The activation function used through the network is the Rectified Linear Unit function (ReLU), which can be thought as a simple if-else condition where if the input is greater than 0 return the input, else return 0. This simple function had a great impact on the training of CNNs, allowing for deeper networks due to reducing the possibility of a vanishing gradient. It also allowed the network to learn complex non-linear relationships in the data, and because it was such a simple function, it greatly improved the speed of which a network was trained. A max pooling layer was used in Figure 5.1 to both, reduce the computational complexity and extract low level features. Max pooling, Figure 5.3, retains the largest value within a define area of the feature maps, which also helps to reduce over-fitting in the network. This section of the network, containing the convolutional layers, can be thought as an unsupervised feature learning (extracting) part. The network learns to extract the most informative features of the data, before it enters the classification stage. Firstly, the extracted features are still in a feature maps format, and must be flattened to become the input to the fully-connected layers. The fully-connected layers are the same as the multi-layer perceptron explained in Section 4.4.2, with a softmax layer attached for the classification output. The softmax function turns the predictions of the network into probabilities that sum to 1, which gives the

predicted class as the biggest probability. Dropout and ridge or L_2 weight regularization is imposed on the network to help regularize it, which helps to counteract overfitting. Dropout randomly drops units from the network, based on an user defined probability. This prevents the co-adaption of features by forcing the network to explore many different paths. L_2 weight decay calculates the L_2 norm of the weights, multiplies it by a regularization strength hyper-parameter, and adds it to the network loss. This pushes the weights to be small, helping to reduce overfitting in the network as large weights can be a sign of an unstable network where small changes in the input can lead to big changes in the output.

5.5.4 AlexNet

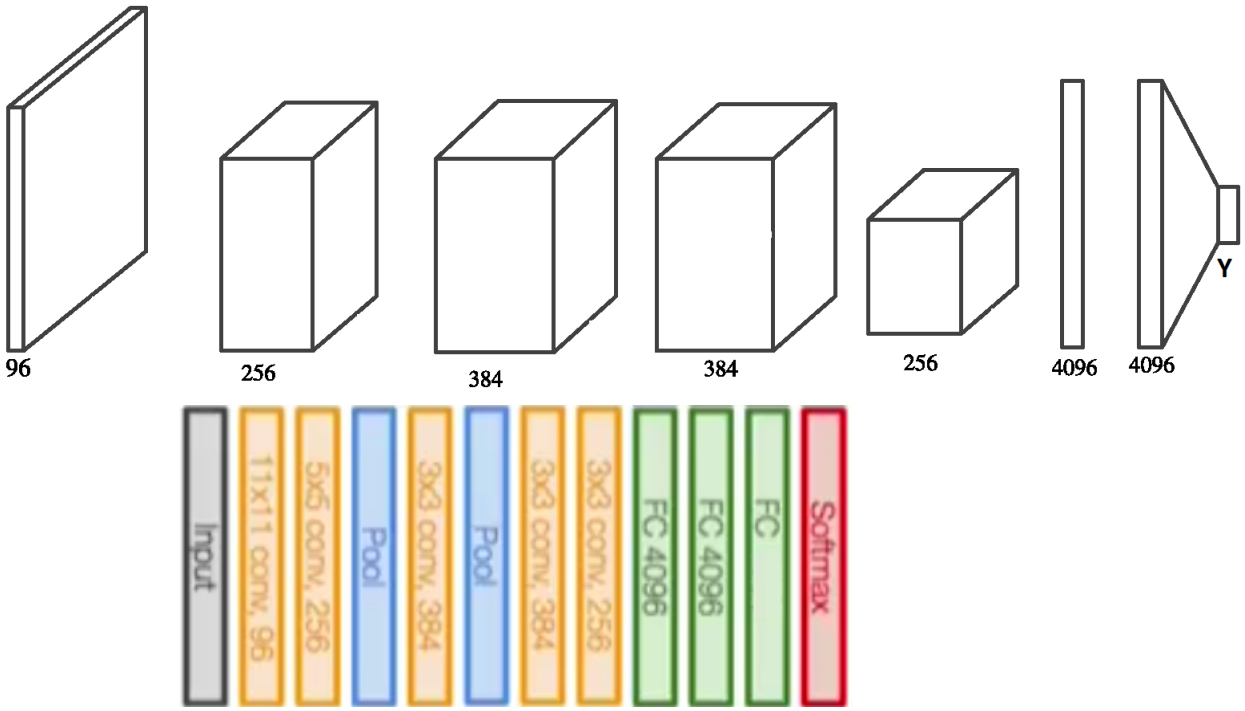

Figure 5.4: Typical architecture and layer sizes of the AlexNet CNN, which consists of 5 convolutional layers followed by 3 fully-connected layers with the size of the last fully-connected layer varying depending on the number of classes (**Y**).

The AlexNet model [KH+09] was the first deep learning approach that made a significant breakthrough in image recognition. It won the 2012 ImageNet ILSVRC challenge by a significant margin. Although based on the neural network LeNet created in 1998 it was deeper and wider, with 5 convolutional layers and 3 dense fully-connected layers. Due to GPU memory restrictions in 2012 they ran the network on 2 parallel GPUs, with inter-GPU connections after the

second convolutional layer, and after all 3 of the fully-connected layers. Unsupervised PulseNet was trained and tested on a single GPU, so instead of using split convolutional layers we created single block layers, as shown in Figure 5.4. AlexNet reduced the classification error on ImageNet by 9.4% from the previous year, but this large improvement came at a cost of computational inefficiency, and our work aims to greatly reduce the model's complexity. AlexNet, being one of the first stand-out deep learning networks, used larger filter sizes and strides to help rapidly reduce the models computational overhead. The first convolutional layer used an 11×11 kernel size with a stride of 4, and the second used a 5×5 kernel size. The rest used filters of size 3×3, and the number of filters throughout the network was 96, 256, 384, 384, 256 followed by two 4096-node fully-connected layers. The last layer of the network was the same size as the number of classes in the dataset. We also use a Rectified Linear Unit (ReLU) as the activation function, and dropout to help counteract over-fitting. Unlike the original network, which used Local Response Normalization, we implement the newer batch normalization technique. The last layer had a softmax activation function instead of the ReLU function, to turn the output predictions into probabilities summing to 1.

5.5.5 VGG16

The next network we test our method on is VGG16 introduced by [SZ14] in 2014, which took second place in the ImageNet ILSVRC challenge for that year. The network had a top-5 test error of 7.3%, which was an improvement of 4.4% from the previous year and outperformed AlexNet by 9.1%. Unlike AlexNet it used a 3×3 filter in all convolutional layers, with stride 1. VGG16 was made up of 13 convolutional layers of sizes 64, 64, 128, 128, 256, 256, 256, 512, 512, 512, 512, 512, 512, two dense fully-connected layers of size 4096, and a final fully-connected layer whose size was the number of classes within the dataset (see Figure 5.5). To reduce network complexity, and also help to spatially generalise it, VGG16 used 2×2 max-pooling after convolutional layers 2, 4, 7, 10, 13. Again the activation function used in all layers except for the last layer, where softmax was used, was ReLu; and batch normalization was used in all layers expect the final layer. Like AlexNet, although the network had an excellent image recognition classification accuracy it was computationally expensive, and we will show that Pulse-Net can greatly improve upon it.

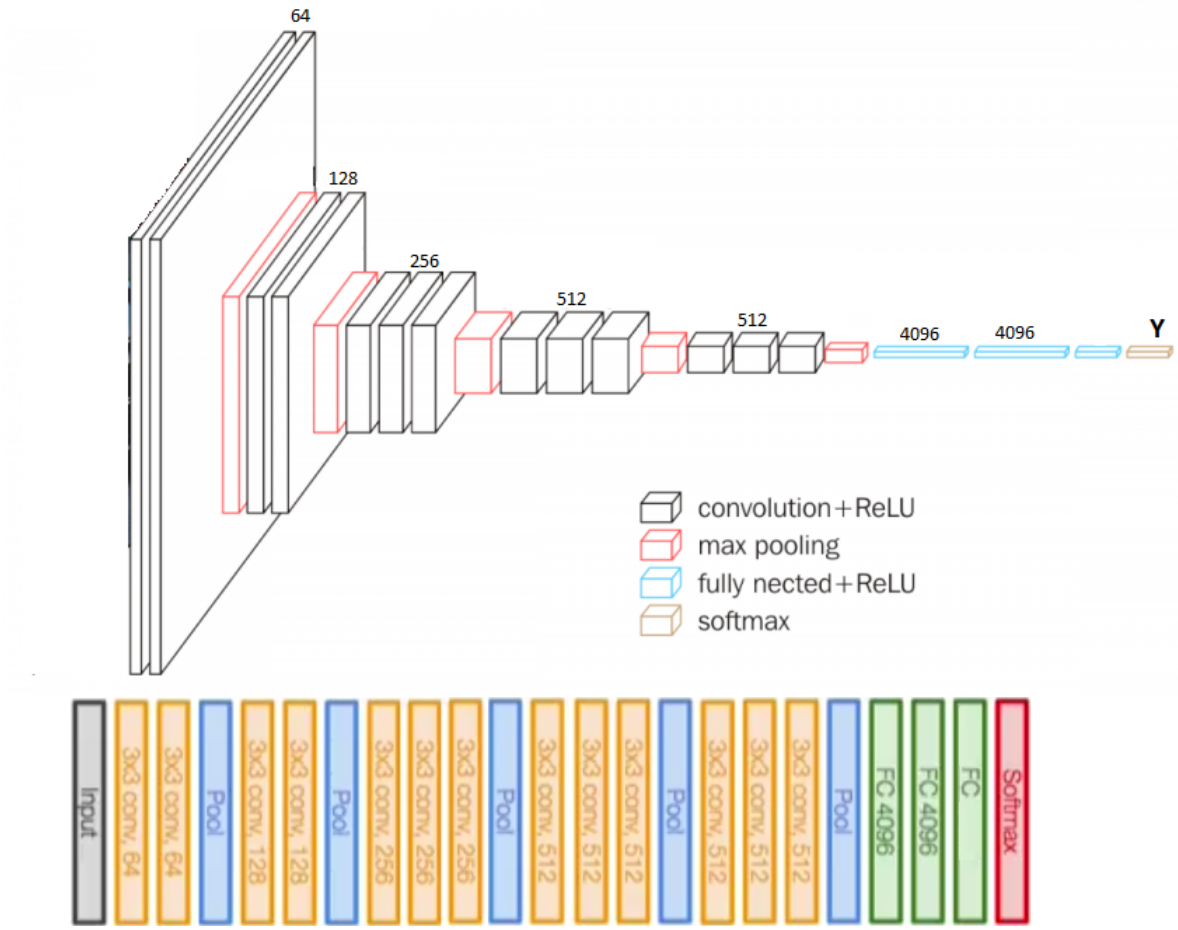

Figure 5.5: Typical architecture and layer sizes of the VGG16 CNN, which consists of 13 convolutional layers followed by 3 fully-connected layers with the size of the last fully-connected layer varying depending on the number of classes (**Y**).

5.5.6 CifarNet

The CifarNet network was a compact CNN, proposed by the TensorFlow group to demonstrate the CIFAR10 dataset using their deep learning framework. It consisted of 2 convolutional layers, with filter sizes 5×5, and had 64 filters in both layers. These were followed by 3 dense fully-connected layers, of sizes in order of 384, 192 and the final size being the number of classes in the dataset. It used 2×2 max-pooling after both convolutional layers, and ReLU activation functions throughout. Batch normalization was also used after each layer of the network. The final layer had softmax as its activation function for classification purposes.

5.6 Results

This section is broken down into 3 subsections, one for each dataset. In each subsection we will compare the accuracy, multiply-add operations (MACs), storage, computational speed and energy efficiency of the compressed and uncompressed networks. Note that an accuracy loss threshold of 2% was enforced during the pruning, meaning the network was allowed to decrease in accuracy by this amount during training to prune the network to a highly compressed state. The results in this section of the accuracy and performance of each test is carried out on a single image put through the network, using the experimental design described above.

Tables showing the statistics for the adversarial images, from each dataset, that required the minimum and maximum amount of noise added can be seen in Tables 5.3, 5.6 and 5.9. The statistics used were (i) noise range which was the range in the pixel values added to the original image to convert it to an adversarial image, (ii) the IQR is the inter quartile range of the noise added, (iii) the mean of the added noise values, (iv) the standard deviation of the added noise values (v) the number of the added noise values classed as outliers, (vi) the mean squared error of the noise added, (vii) the structural similarity is a measurement of the similarity between the original image and the adversarial image, and (viii) entropy is the joint entropy between the original image and the adversarial image.

5.6.1 CIFAR10

It can be seen from Table 5.2 that Pulse-Net was able to reduce AlexNet by 95.63% and VGG16 by 87.85%, sacrificing only 2% and 0.9% accuracy respectively. CifarNet was pruned by 76.25% with a decrease in accuracy of 3.6%. It is worth remembering that this model was designed specifically to demonstrate image recognition tasks using the CIFAR datasets, and was already an optimally sized network. The relevant filters and nodes of these pruned networks were extracted and reloaded onto a network suited to their size. These reduced networks were then tested for efficiency, displaying substantial savings in storage and up to 50% reduction in inference time, with a saving of between 14% and 65% in energy required for inference.

These extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics from Table 5.3 describe the noise

Table 5.2: The table shows the performance and accuracy of the original CifarNet, AlexNet and VGG16 networks, using the CIFAR10 dataset, and are compared to the Pulse-Net compressed versions, where the percentages are the percentage reductions.

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Inference Time (ms)	Energy (mJ)
AlexNet	91.16	5.83 X 107	874	222.5	4.14	0.95
Compressed	2%	95.63%	95.31%	95.63%	50.72%	65.26%
CifarNet	85.36	1.07 X 106	18.4	4.08	1.69	0.39
Compressed	3.6%	76.25%	72.83%	76.25%	2.95%	13.59%
VGG16	90.87	3.36 X 107	287.2	128.36	6.32	0.84
Compressed	0.9%	87.85%	87.89%	87.85%	47.94%	59.52%

added to the images that required both the maximum and minimum amount of variation needed to create the adversarial images to fool the network. The images where most noise was added were deer (2995), airplane (7835) and frog (1578), while the images requiring the least amount were airplane (2473), ship (3140) and car (6010) on the networks Adv-AlexNet, Adv-CifarNet and Adv-VGG16, respectively. It can be seen that there was a great difference between the amount of noise required between creating the max and min adversarial images. This is highlighted in the results of the MSE statistic and the R-Squared values, along with the noise range values.

Table 5.3: Statistics of the CIFAR10 adversarial images that required the maximum and minimum amount of noise added.

	Adv-AlexNet		Adv-CifarNet		Adv-VGG16	
Statistic	Maximum	Minimum	Maximum	Minimum	Maximum	Minimum
Noise range	[-72, 161]	[-12, 15]	[-82, 82]	[-12, 13]	[-158, 151]	[-14, 12]
IQR	11	0	2	0	14	2
Mean	0.23	0.009	-0.18	-0.004	0.05	0.04
STD	21.45	2.03	11.96	2.1	18.46	2.5
Outliers	655	1420	717	1361	296	250
MSE	1381.04	13.32	428.98	13.29	1021.94	18.71
Structural Similarity	0.27	0.98	0.72	0.98	0.8	0.98
Entropy	5.86	2.52	3.83	2.5	5.83	3.12

Table 5.4 shows accuracy results for the adversarial attacks using the CIFAR10 dataset. Taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unpruned models, the table shows that out of the 9 attacks, 5 of Pulse-Net's pruned models were more robust than the original models. This illustrates that for the CIFAR10 dataset Pulse-Net created networks were not only more efficient but also over 56% more robust to adversarial attacks (see Table 5.4), meaning it is harder to fool the compressed network with adversarial images than it is to fool the original network with the adversarial images. The higher the accuracy in this table,

the more likely the networks were to classify an adversarial image as a true image, therefore the lower the accuracy the better. We believe the reason for the compressed network being more robust against adversarial attack is due to the reduced number of parameters in the network which can be tricked under the attack.

Table 5.4: Accuracy results of CIFAR10 adversarial images created by CifarNet, AlexNet and VGG16.

	Adv-AlexNet		Adv-CifarNet		Adv-VGG16	
Network	Original (%)	Compressed (%)	Original (%)	Compressed (%)	Original (%)	Compressed (%)
AlexNet	20.08	20.74	31.61	20.51	42.69	35.11
CifarNet	37.90	39.49	9.89	9.38	45.94	44.21
VGG16	40.92	43.68	37.54	36.06	29.26	29.34

5.6.2 CIFAR100

It can be seen from Table 5.5 that Pulse-Net was able to reduce AlexNet by 85.95% and VGG16 by 87.6%, with only a loss in accuracy of 2.3% and 1.1% respectively. CifarNet was pruned by 65.57% with a decrease in accuracy of 4.5%. Again, this network was already optimised. After extraction these reduced networks were tested for efficiency, and like the CIFAR10 results showed great improvement in storage, reducing the inference times by between 0.6% and 48.33%, and with a saving of energy between 9% and 67%.

Table 5.5: The table shows the performance and accuracy of the original CifarNet, AlexNet and VGG16 networks, using the CIFAR00 dataset, and are compared to the Pulse-Net compressed versions, where the percentages are the percentage reductions.

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Inference Time (ms)	Energy (mJ)
AlexNet	69.77	5.87 X 107	874.5	223.9	4.16	1.04
Compressed	2.31%	85.95%	85.76%	85.95%	46.15%	67.46%
CifarNet	58.05	1.09 X 106	18.5	4.14	1.68	0.40
Compressed	4.5%	65.57%	63.03%	65.57%	0.6%	8.79%
VGG16	64.2	3.4 X 107	287.6	129.77	6.29	0.83
Compressed	1.14%	87.6%	87.87%	87.6%	48.33%	57.52%

Following the same testing method as for the CIFAR10 dataset, these extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics from Table 5.6 describe the noise added to the images that required both the maximum and minimum amount of variation needed to create the adversarial images to fool the network. The images where the most noise was added were cloud (8703), ray (7318) and rocket (1221), while the images requiring the least amount were kangaroo (5012), whale (1727)

and turtle (628) on the networks Adv-Alexnet, Adv-CifarNet and Adv-VGG16, respectively. The statistics for the CIFAR100 dataset follow a similar pattern to those for CIFAR10.

Table 5.6: Statistics of the CIFAR100 adversarial images that required the maximum and minimum amount of noise added.

	Adv-Alexnet		Adv-CifarNet		Adv-VGG16	
Statistic	Maximum	Minimum	Maximum	Minimum	Maximum	Minimum
Noise range	[-71, 89]	[-18, 12]	[-46, 162]	[-25, 22]	[-148, 107]	[-20, 22]
IQR	10	2	5	2	13	4
Mean	-0.006	-0.006	0.4	0.002	-0.02	0.015
STD	12.7	2.29	6.99	3.64	17.75	4.09
Outliers	282	193	234	526	358	179
MAE	8.39	1.35	3.91	2.38	11.26	2.79
MSE	484.14	15.7	147.05	39.66	945.41	50.06
RRMSE	0.09	0.016	0.15	0.036	0.12	0.05
Structural Similarity	0.535	0.996	0.41	0.981	0.29	0.95
R Squared	0.704	0.998	0.983	0.993	0.36	0.993
Entropy	5.46	2.84	4.33	3.68	5.77	3.91

Table 5.7 shows the accuracy results from the adversarial attacks using the CIFAR100 dataset. Again, taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unpruned models, the table shows that out of the 9 attacks, 5 of Pulse-Net's pruned models were more robust than the original models. This illustrates that for the CIFAR100 dataset Pulse-Net created networks were not only more efficient but were also nearly 56% more robust under adversarial attacks (see Table 5.7).

Table 5.7: Accuracy results of CIFAR100 adversarial images created by CifarNet, AlexNet and VGG16.

	Adv-AlexNet		Adv-CifarNet		Adv-VGG16	
Network	Original (%)	Compressed (%)	Original (%)	Compressed (%)	Original (%)	Compressed (%)
AlexNet	14.90	17.25	18.48	15.84	32.38	30.10
CifarNet	26.47	26.03	4.14	4.28	26.92	25.22
CifarNet	29.35	30.71	20.56	20.13	19.23	22.29

5.6.3 Tiny-ImageNet

Finally, looking at Table 5.8, Pulse-Net was able to reduce AlexNet by 80.79% and VGG16 by 83.15%, with only a loss in accuracy of 3.75% and 2% respectively. CifarNet was pruned by 74.25% with a decrease in accuracy of 2.91%. The reduced extracted networks were tested for efficiency, and like both CIFAR results showed great improvement in storage, reduced the inference times by between 15% and 57%, and a saving of energy between 13% and 65%.

Table 5.8: The table shows the performance and accuracy of the original Cifar-Net, AlexNet and VGG16 networks, using the Tiny-ImageNet dataset, and are compared to the Pulse-Net compressed versions, where the percentages are the percentage reductions.

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Inference Time (ms)	Energy (mJ)
AlexNet	54.8	7.27 X 107	1266.2	277.47	8.68	1.32
Compressed	3.75%	80.79%	79.20%	80.79%	56.57%	65.15%
CifarNet	40.45	5.04 X 106	100	19.22	2.1	0.31
Compressed	2.91%	74.25%	71.26%	74.25%	15.24%	12.9%
VGG16	56.05	4.07 X 107	1010.8	155.33	6.67	0.86
Compressed	2%	83.15%	83.81%	83.15%	44.68%	61.63%

Following the same testing method as the CIFAR datasets, these extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics from Table 5.9 describe the noise added to the images that required both the maximum and minimum amount of variation needed to create the adversarial images to fool the network. The images where the most noise was added were flagpole (63), pole (7748) and stopwatch (5861), while the images requiring the least amount were poncho (707), tractor (4392) and teddy (895) on the networks Adv-Alexnet, Adv-CifarNet and Adv-VGG16, respectively. The statistics for the Tiny-ImageNet dataset show that less noise was required to create adversarial images, as shown in the MSE and structural similarity statistics.

Table 5.9: Statistics of the Tiny-ImageNet adversarial images that required the maximum and minimum amount of noise added.

	Adv-Alexnet		Adv-CifarNet		Adv-VGG16	
Statistic	Maximum	Minimum	Maximum	Minimum	Maximum	Minimum
Noise range	[-91, 67]	[-9, 9]	[-59, 143]	[-23, 19]	[-72, 63]	[-9, 12]
IQR	6	2	2	4	5	2
Mean	-0.015	0.014	0.07	0.03	-0.04	-0.0007
STD	7.29	1.44	4.78	4	8.92	1.42
Outliers	980	116	1822	644	2081	166
MAE	2.86	0.96	2.45	2.76	5.13	0.86
MSE	83.72	6.19	68.58	47.96	238.76	6.07
RRMSE	0.03	0.02	0.03	0.04	0.07	0.01
Structural Similarity	0.919	0.996	0.92	0.991	0.997	0.997
R Squared	0.994	0.999	0.996	0.996	0.992	0.999
Entropy	3.6	2.48	3.69	3.92	4.54	2.36

Table 5.10 shows the accuracy results from the adversarial attacks using the Tiny-ImageNet dataset. Again, taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unpruned models, the table shows that out of the 9 attacks, 8 of Pulse-Net's pruned models were more robust than the original models. Again, displaying

that for the Tiny-ImageNet dataset Pulse-Net created networks were not only more efficient but were nearly 89% more robust to adversarial attacks (see Table 5.10).

Table 5.10: Accuracy results of Tiny-ImageNet adversarial images created by CifarNet, AlexNet and VGG16.

	Adv-AlexNet		Adv-CifarNet		Adv-VGG16	
Network	Original	Compressed	Original	Compressed	Original	Compressed
AlexNet	40.93	36.99	39.15	33.77	47.29	42.49
CifarNet	32.79	30.88	16.02	15.95	34.86	32.61
VGG16	46.89	45.34	43.16	41.77	30.49	35.54

5.7 Conclusion

In this chapter, we proposed a novel deep CNN pruning method called Pulse-Net, which compresses a network during training to create a more efficient model for inference. The proposed compression method shows significant improvements in storage, inference timings and energy efficiency, as well as greater robustness under adversarial attack, all of which is ideal for IoT devices. This chapter laid the fundamental building blocks of a novel approach of pruning CNNs with the aim of increasing their overall efficiency.

Pulse-Net showed us that CNNs can be pruned successfully in an iterative manner, and although the filters/nodes to be removed were selected using a rather simple minimum value decision, we were able to significantly prune the networks with negligible loss in classification accuracy. The networks pruned by Pulse-Net had many application purposes including mobile phone CNNs, smart cameras, as well as assisting self-driving cars.

In the next chapter, Chapter 6, we will make significant advancements on the idea, which we show gives us a better compression-to-accuracy ratio. We will also take an idea from RRD (see Section 2.5.2) and iRRD-DGA (see Section 4.5.2), removing the need of user defined pruning rate, making it an unsupervised method. Another great benefit we will introduce with the new method is a speed-up in the training process. This is achieved by extracting a smaller network at each pruning iteration, instead of using a binary matrix to simulate filters/nodes being removed.

Chapter 6

Unsupervised PulseNet: Automated Pruning of Convolutional Neural Networks by K-Means Clustering

6.1 Motivation

Following on from the success of Pulse-Net (see Section 5.4), we look at ways to improve the algorithm in this chapter. The motivation behind the algorithm is still to develop very efficient (computation and energy-wise) networks that can be deployed on mobile devices with limited memory and power. A very active area of research is still the compression of deep networks while maintaining accuracy, with the aim of reducing memory usage, energy consumption and processing time. Several network compression methods have been proposed and have given good results, but they usually require the specification of parameters and are computationally expensive. We propose a new fast automated method called Unsupervised PulseNet that uses unsupervised k-means clustering to detect clusters of similar filters, and nodes in fully-connected layers, and prunes those that are redundant. We evaluate it on the CIFAR10, CIFAR100 and Tiny-Imagenet datasets using Alexnet, VGG16 and a 2-layer CNN called CifarNet suggested by the Tensorflow group. Compared to other methods in the literature we achieve the greatest compression, in shorter times, and with negligible loss in classification accuracy.

6.2 Introduction

CNNs have additional fully-connected layers at the top, and most work on network pruning focuses on reducing either the convolutional or the fully-connected layers. We decided to prune both convolutional filters and fully-connected layer nodes, thus aiming for high compression. This pruning is independent of several other compression methods mentioned above (sparsity, weight binarization and network quantization) and can potentially be combined with them. It also does not require additional software or hardware so it is easy to deploy in the field. To achieve this pruning we propose a new iterative algorithm called *Unsupervised PulseNet* that finds optimal clusters within the filters or nodes of each layer, selecting only those nearest to the centroids (approximate *medoids*), along with all their associated parameters, and discarding the rest of each cluster. Retraining the reduced network usually recovers any loss in accuracy, and reiterating this pruning and fine-tuning process until the accuracy cannot be improved further achieves a high degree of compression with little loss in accuracy. If the network has been pruned beyond recovery, we can easily revert back to the best state found so far, and either prune less aggressively [BGP19] or move onto the next layer for pruning.

Extensive experiments were performed on various datasets and different neural networks, to evaluate the performance and robustness of Unsupervised PulseNet. The experiments are on computer vision tasks, but the method is also applicable to other deep learning problems such as object detection based on CNNs or image segmentation using Auto-Encoders. The results show that Unsupervised PulseNet is significantly better than other methods, achieving the best compression rate in the literature: it compresses VGG16 by $21\times$ with 2.5% loss of accuracy; AlexNet by $147\times$ with 1.7% loss of accuracy; and CifarNet by $5.4\times$ with 2.5% loss of accuracy. The main contributions of this research can be summarized as follows:

- We introduce a novel approach, called Unsupervised PulseNet, that automatically selects redundant filters/nodes in a CNN, removes them and compresses the model.
- The method is the first to use an unsupervised algorithm (K-means clustering) to automatically cluster similar filters and nodes, greatly reducing training computation times.
- It achieves state-of-the-art results, pruning VGG16, AlexNet and CifarNet

models into 6.07MB, 1.21MB and 0.57MB respectively, with little loss in classification accuracy.

The rest of the chapter is organised as follows. Section 6.3 discusses related work on CNN pruning and compression. Section 6.4 explains our proposed method, and the datasets and CNNs used in the experiments. Section 6.5 presents and discusses the results. Section 6.6 concludes the chapter.

6.3 Related Work

[IHM+16, HZC+17] designed CNNs that were much smaller than standard networks yet achieved similar accuracy. MobileNets [HZC+17] use depth-wise separable convolutions to create a lightweight CNN, and by allowing the user to adjust hyper-parameters they can create an efficient and accurate model. SqueezeNet [IHM+16] uses a bottleneck approach to create a very small network, replacing the majority of its 3×3 filters by 1×1 filters, and also reducing the number of input channels to the remaining 3×3 filters. The authors allow the convolutional layers to have large activation maps, by down-sampling towards the end of the network. This helps to maximize accuracy, while adjusting the filters in the network aims to reduce the number of model parameters. ShuffleNet was designed to be extremely efficient on mobile devices [ZZLS18]. Its architecture makes use of both point-wise group convolutions and channel shuffling to greatly decrease computation cost with little loss of accuracy. [YYX+18] proposed using width multipliers, but instead of training separate networks with defined widths they trained a shared network with switchable batch normalization. This allowed the network to adjust its width based on the device constraints.

[YLC+18] formulated their pruning method using a binary integer programming problem and gave a closed-form approximate solution. Their solution took into account not only a singular weight, but also the inputs and other nodes in the network. Their work differed to ours mainly in that they measured the value of the nodes across the network, while we evaluated them layer-by-layer. [CHS+16] quantized the weights in the network with binary values. [RORF16] proposed using XNOR and bit-count operations in the convolutional layers, which performed quantization on their weights in a network model called XNOR networks. [HMD15, MLDD17] also exploited pruning, quantization and Huffman coding to compress CNNs. [CHS+16] proposed

using the gradient with respect to the quantized parameters together with randomized rounding method when training a model from scratch. It should be noted that these methods can complement filter pruning, and therefore can be performed in conjunction with Unsupervised PulseNet for compression rate improvement.

[LKD+16] focused on pruning entire convolutional filters. The pruning metric used was the L1 norm of all the weights within the filter. The filters were then ranked according to their L1 value, and the n lowest ranked filters layer-wise were pruned. The value n was chosen to be a predetermined number or percentage at each iteration to prune the network. The network was then fine-tuned to regain accuracy, and the whole process was repeated until the loss in accuracy was too great to recover from. One of their experiments was an analysis of VGG16 using the CIFAR-10 dataset, which can be directly compared to our results. They reduced the network to 64% of the original number of parameters, whereas we reduced it to 4.7% while maintaining similar accuracy.

[CTSO18] compared the effects of pruning a network followed by fine-tuning (the approach taken in our work) to pruning the network followed by retraining from scratch. They tested two pruning methods — L1-norm [LKD+16] and Fisher pruning [MTK+16] — and found promising results when the smaller networks were trained from scratch.

[BGP19,ZRZ+18] use a simple pruning method that analyzes the networks feature maps. When the network is fully trained [BGP19] used the L1 magnitude of each feature map to rank their corresponding filters, removing a predefined number of the lowest ranking ones. While [ZRZ+18] also used feature maps, their pruning decision was based on linear discriminant analysis. Both methods retrained the pruned networks to retrieve as much accuracy as possible. The experimental results of [BGP19] can be directly compared to this work, as can those of [ZRZ+18] on VGG16 using CIFAR10. [HLW+19] argued that the geometric median would be a more appropriate metric than L1 for pruning purposes. The geometric median is a well-known robust estimator of centrality in Euclidean space, and according to its characteristics, filters close to the median, can be represented by the remaining filters, hence chosen to be pruned.

[LWL17] proposed ThiNet that used a greedy approach to prune channels with the least effect on the following layer's activation values. [HKD+18], on the other hand, pruned channels by looking at the next network layer's feature maps and minimizing their reconstruction error using LASSO regression. Both

found that training the pruned network from scratch caused a reduction in accuracy. [LSZ+18] implement both their methods, and when they allowed the network to train from scratch but for a longer period, they found that it was possible to achieve slightly better accuracy than the fine-tuning approach.

[LAT18] introduced a single-shot network pruning approach that created an extremely sparse network. Their method identifies the structurally important connections in the network, based on their influence on the loss function at initialization prior to training, namely connection sensitivity. Using a specified pruning rate, redundant connections are pruned before training begins. This means that, instead of taking a pre-trained network and pruning it followed by fine-tuning, their method prunes the network and then trains it, hence the name "single-shot pruning". Their approach prunes individual connections and not filters or nodes, and therefore needs additional software to see any actual speed up at inference time, whereas we prune the network and end up with a slim version of the original network.

[DCP17] adopted a layer-wise pruning approach. They removed parameters in each layer using a second order derivative metric of the layer's error function. A benefit of this approach is that the network required only limited fine-tuning, although performing a layer-wise reduction can be computationally expensive. To help improve efficiency they suggest a simplified way of calculating the Hessian matrix, thus speeding up the computation of its inverse. Their experiments include the use of CIFAR-10 dataset and the same networks used in this chapter (VGG-16, AlexNet and CIFAR-Net) so we can compare compression rates achieved between both algorithms.

[HZYN18] formulated a reinforcement learning method called try-and-learn that learned which filters to prune. An agent is trained to determine which filters to prune and which to retain. As this pruning is carried out in an automated and data-driven method, and the agent's reward function allows a trade-off between accuracy and network size, their approach gives high compression and good accuracy. This is also true of Unsupervised PulseNet as during pruning there is no involvement from humans in the loop, and as our results below show, we outperform them.

[DDHT18] combined two compression ideas: firstly, constrain the network during training leading to structured sparsity [WWW+16]; secondly, iteratively prune and retrain filters [BGP19]. The intuition behind this approach is that by initially forcing parameters to be close to zero, when pruning complete fil-

ters the network will suffer less damage. For the first idea they use a slightly customized l_2 norm to penalize weak filters, then use the l_1 norm as the filter importance metric to determine which filters to remove in the second part. We will compare their results on the CIFAR-10 dataset in Section 6.5.

6.4 Materials and Methods

The robustness and performance of Unsupervised PulseNet was demonstrated using two state-of-the-art CNNs AlexNet and VGG16 and a CNN called CifarNet developed specifically for the CIFAR10 dataset. It was evaluated on three benchmark datasets CIFAR10, CIFAR100 and Tiny-Imagenet.

This section is broken down into three subsections: an outline of the experimental design, then a description of the classifiers and datasets used, and finally an explanation of our proposed algorithm.

6.4.1 Experimental Design

To develop Unsupervised PulseNet we used the Python programming language and the TensorFlow deep learning framework. The training and pruning phases were performed on a GTX1080 NVIDIA graphics processing unit (GPU), while the inference stage was evaluated on both the GPU and on an INTEL I7-7777 CPU with 16GB RAM.

For a fair comparison with other related work we used standard data augmentation, including random horizontal flipping in which a training image was reflected about a vertical axis with probability 0.5. Another augmentation used was random adjustments of the image brightness, where a random value between a positive and negative threshold (± 63 in this work) was selected and used to adjust the pixels of an image. The final augmentation used was a random adjustment of the image contrast (the difference between the maximum and the minimum pixel intensity in the image), using a random value selected between two thresholds (0.2 and 1.8 for this work). The optimizer used was stochastic gradient descent (SGD) with the learning rate reduced via a stepwise decay learning rate schedule. The initial learning rate (see Section 5.4) for both training and fine-tuning was 0.1, which was reduced by a factor of 10 when no decrease in the loss on the validation set was detected for 20 epochs. The minimum learning rate used was 0.00001, and once this was reached and

no further improvement in the loss was found for 20 epochs, the network is considered to have converged. Batch normalization was used before applying the activation function in all network layers except the output layer. The image inputs were divided by 255 to bound all the data between 0 and 1.

6.4.2 Convolutional Neural Networks & Datasets

In this chapter, we use AlexNet, VGG16 and CifarNet, explained in Sections 5.5.4, 5.5.5 and 5.5.6, respectively, as the CNNs to prune to demonstrate the robustness of our proposed approach. Both AlexNet and VGG16 have shown great success in image classification tasks, and CifarNet is an already small network designed by Tensorflow specifically for the CIFAR10 dataset.

Again, we use 3 benchmark datasets within the image recognition research area, CIFAR10, CIFAR100 and Tiny-ImageNet to evaluate the performance of Unsupervised PulseNet. We have given a detailed description of them in Section 5.5.2.

6.4.3 Unsupervised PulseNet

Unsupervised PulseNet is a CNN compression algorithm that takes a trained network and iteratively, using a k-means based approach to cluster the filters and nodes separately within each layer, prunes filters and nodes throughout the network, then fine-tunes the model to regain accuracy. The algorithm continues this process on all layers until the loss in accuracy breaches a threshold (we chose 2% for this work), then the network reverts back to its best previous state (the state at which the network was able to recover it's accuracy within the threshold). At this point, instead of trying to prune all layers in one iteration, the network is pruned in sections, starting with the fully-connected layers as these contain the majority of the network parameters. Again, once the accuracy threshold is exceeded these layers revert to their best previous state (all layers rolled back to the last state where the network's accuracy was within the threshold), then the convolutional layers are pruned (these are the most computationally expensive part of the network). Finally, for the last breach of accuracy, each layer is individually pruned, iteratively.

This type of pruning approach can be thought of as coarse (entire network), then medium (each section; convolutional and fully-connected), then fine pruning (individual layers): we believe that this incremental approach helps us to

achieve state-of-the-art compression results for the different datasets, as shown in Section 6.5. We show that by using a more intelligent way of selecting the filters or nodes considered to be redundant, which are then pruned from the network, we can achieve an extremely efficient CNN.

The following bullet points give a line-by-line description of Algorithm 7:

- In line 1 we define γ and ρ as the convolutional and fully-connected layers of the network.
- In line 2 we define the network classification accuracy, on the validation set as λ.
- In line 3 we initialize i (layer number) to 0, β (stopping criteria as threshold for validation accuracy loss) to 2 and L to $\gamma + \rho$ (indicating the algorithm is analyzing both the convolutional and fully-connected layers of the network).
- Lines $4 - 41$ are a loop; this is exited (line 40) when the algorithm has produced a fully pruned network..
- This loop is the main part of the algorithm, and is where the algorithm selects which filters/nodes to remove.
- In line 5 we initialize p_{in} to represent the 3 channels of an RBG image.
- Lines $6-24$ are a loop where we loop through all the layers of the network.
- In line $7 - 13$ is an if condition checks if we are analysing a convolution layer.
- Line 8 defines the filter matrix of the layer as a 4-D matrix of shape w, x, y, z.
- In line 9, w and x is the shape of the kernal in the layer (3×3), and y represents the input shape and z is the output shape. We remove the rows in y that are not in p_{in}.
- Line 10 we reshape the 4-D matrix into a 2-matrix without changing its data.
- Line 11 using Algorithm 8 to return the index of important rows, which are the clusters centers, as p_{out}.
- In line 12 we remove the rows in z that are not in p_{out}.

- Line 13 we reshape the 2-D matrix back into a 4-matrix without changing its data.
- In line $15-22$ is an if condition checks if we are analysing a fully-connected layer.
- Line 16 defines the node matrix of the layer as a 2-D matrix of shape $m,\ n$.
- In line 17 we remove the rows in m that are not in p_{in}.
- Line 18 we transpose the 2-D matrix by rotating it around its axis, without changing its data.
- Line 19 using Algorithm 8 to return the index of important rows, which are the clusters centers, as p_{out}.
- In line 20 we remove the rows in n that are not in p_{out}.
- Line 21 we transpose the 2-D matrix back into its original shape by rotating it around its axis, without changing its data.
- In line 23 the desired output cluster centers of the previous layer becomes the input clusters of the following layer. Therefore we set p_{in} to p_{out}.
- Line 25 we fine-tune the network to convergence.
- In line 26 we calculate the new classification accuracy on the validation set, as δ.
- Lines $27-40$ which checks to ensure the network maintains classification acuracy within the threshold limit β.
- Lines $28-32$ are if conditions that check what layer type is under analysis, and switches it to the next type.
- Lines $33-38$ checks if all the layers have been fully pruned, and if so stops the algorithm. If they have not all been fully pruned it moves onto the next layer.

Algorithm 7 Unsupervised PulseNet Algorithm

Define γ and ρ as the network's convolutional and fully-connected layers
Define λ as the classification accuracy of the validation set
Initialize $i = 0,\ \beta = 2\%,\ L = \gamma + \rho$
while $i \neq$ (len(γ)+ len(ρ)) **do**
    initialize set $p_{in} = \{0, 1, 2\}$
    for all $l \in L$ **do**
        if $l ==$ convolution layer **then**
            given filter F represented by (w, x, y, z)
            Remove rows in y not in p_{in}
            Transpose $(w, x, y, z) \rightarrow (z, w * x * y)$
            p_{out} =Get optimal k clusters of $(z, w * x * y)$ (Algorithm. 8)
            Remove rows in z not in p_{out}
            Transpose $(z, w * x * y) \rightarrow (w, x, y, z)$
        end if
        if $l ==$ fully connected layer **then**
            given node N represented by (m, n)
            Remove rows in m not in p_{in}
            Transpose $(m, n) \rightarrow (n, m)$
            p_{out} =Get optimal k clusters of (m, n) (Algorithm. 8)
            Remove rows in n not in p_{out}
            Transpose $(n, m) \rightarrow (m, n)$
        end if
        $p_{in} \leftarrow p_{out}$
    end for
    Fine-Tune Network until it converges
    Calculate δ = new validation accuracy
    if $(\lambda - \delta) > \beta$ **then**
        if $L == \gamma + \rho$ **then**
            $L = \rho$
        else if $L == \rho$ **then**
            $L = \gamma$
        else $[L == \gamma]$
            if $i ==$ (len(γ)+ len(ρ)) **then**
                Halt
            else
                $L = i^{th}$ layer of the full network
                $i = i + 1$
            end if
        end if
    end if
end while

Unsupervised PulseNet uses an unsupervised k-means clustering algorithm, and a maximum pruning factor of 25% was imposed: although k-means automatically determines the number of clusters in the data, it was limited to only looking at a quarter of the layer at a time. The number of clusters, the k value, was automatically selected using an automated version of the elbow rule [KM13]. The maximum pruning factor was chosen to be 25%, as we believe it gave the algorithm the opportunity to pruned enough of the layers without causing an unrecoverable loss in accuracy. This helped to reduce the possibility of the network suffering unrecoverable *brain-damage* (a state where the network is unable to recover the loss in accuracy within a threshold), and allowing Unsupervised PulseNet to compress the network even further.

The convolutional layers parameters are represented by w, x, y, z, where w and x are the filter sizes, y is the number of inward filters and z the number of outward filters. When analysing the fully-connected layers, since there are no filters, the layer is represented by m which is the number of inward nodes and n, the number of outward nodes. We call filters-in the the list of inwards filters indices, and filters-out the list of outward filters indices, from the above tuneable parameters. It should be noted that if we are pruning the first convolutional layer, then the filters-in would be the colour depth of the input image, which for an RGB image would be 3 inward filters represented by the list of filters $\langle 0, 1, 2 \rangle$. For the remaining convolutional layers, the filters-in are the filters-out from the previous layer.

To find the filters-out, Unsupervised PulseNet clusters the outward filters using Algorithm 7, retaining only the filter nearest to the centroid of each cluster, removing the rest which are deemed to be redundant. The retained filter indices or the filters-out become the next layer's filters-in and are used to subset its inward filters. This is repeated for all the layers in the network. The filters-out are found by running the k-means algorithm with all possible k values between a quarter of the number of filters/nodes in the layer, s and the total number of filters/nodes in the layer. As stated above, this puts a pruning limit in each layer to help the network not suffer unrecoverable accuracy loss. The sum of the L2 distances between each filter/node and the centre within each cluster (distortion) is calculated. The algorithm runs until either no change, within a very small tolerance, of the distortion occurs or the maximum number of iterations is reached (300). The *k-means* algorithm had 5 restarts to reduce the possibility of getting stuck in a local minimum. The distortion is recorded, along with its corresponding k value. By using Algorithm 8 we determine the

optimal number of filters/nodes to retain. Algorithm 7 shows how it loops through the layers of the network, pruning each layer in an unsupervised way. The relevant filters or nodes that Unsupervised PulseNet extracts are re-used to create a smaller network, and the network is fine-tuned on the dataset to regain any loss in accuracy.

Algorithm 8 Get optimal k clusters

Set $\alpha = 0.25$
given input M
$s = \alpha \times \text{len}(M)$
for all k in $s \rightarrow \text{len}(M)$ **do**
    k random vectors V as initial clusters centres (ICC)
    repeat
        (re)assign each V to ICC to which it is most similar,
        based on mean value of V within ICC
        Update ICC means
    until *no change*
    Store sum of squares within clusters (WCSS)
end for
$D1 = WCSS_{i+1} - WCSS_i, \quad \forall \quad i \in 1 \rightarrow \text{len}(WCSS) - 1$
$D2 = D1_{i+1} - D1_i, \quad \forall \quad i \in 1 \rightarrow \text{len}(D1) - 1$
$\sigma = D2 - D1$
$P \rightarrow \{\}$
for all u in σ **do**
    if $u > 0$ **then**
        P.insert(u)
    end if
end for
C = index of max(P)
$Y \rightarrow \{\}$
for all centre in C **do**
    v = index of vector closest to centre
    Y.insert(v)
end for
return Y

The following bullet points give a line-by-line description of Algorithm 8:

- Line 1 we set α to 0.25 which tells the algorthim not to try to cluster the first quarter of the number of filters/nodes in the layer. This helps the network to avoid potential unrecoverable loss.
- Lines 2 and 3, takes the matrix from the main algorithm, as M, and calculates s, the starting point of where the algorthim can check for cluster centres.
- Line $4-12$ is a loop which searches for clusters from s to the length of the matrix, which would indicate only single point clusters.
- In line 5 the algorthim initials random vectors as inital centres.
- Lines $6-10$ the algorthim repeatedly searches to optimise the cluster centres.
- In line 11 the sum of squares within clusters are stored (WCSS).
- In lines $13-15$ an automated approach to finding the optimal number of clusters ([KM13]) is used. By calculating the first order differnce between the WCSS, and then the second order differences, σ is computed by getting the difference between them.
- Lines $17-21$ is a loop that continues the automated approach by checking which of the σ values are positive.
- In line 22 the optimal number of clusters C is found by getting the index of the maximum positive σ value from the above line.
- Line 23 initializes an empty set.
- Lines $24-27$ is a loop that gets the index row of all the vectors in the given matrix M, which is closest to each of the centres in C
- Line 28 returns the set of important rows found in the above line, which will be used to reduce the layers in the main algorthim.

We give a higher level explanation of both algorithms, Algorithm 7 and Algorithm 8 to help the reader fully understand the method. We start with the main algorithm, Algorithm 7. We define *i* as the number of the layer the method is looking at (this parameter is only used in the fine pruning part of the algorithm as mentioned above). The parameter λ is the validation accuracy of the network from its initial converged state (which is the original network size). This value is used as a comparison to the validation accuracies as the network is pruned.

Both γ and ρ are the layers of the network, convolutional and fully-connected, respectively. They are the indices of the layers, which the algorithm uses to loop through the layers, and also used to stop the algorithm by adding the number of layers within both to check if all layers have been fine pruned. The final parameter is β, which the user defines (in all experiments in this chapter it is set to 2%) to help control the loss in accuracy. This can be adjusted according to how much accuracy loss the user is willing to accept. As stated above, the algorithm starts with coarse pruning (all the network layers), and in the first layer the set of filters to keep (p_{in}) is set to $\langle 0, 1, 2 \rangle$ (corresponding to the input being a RGB image). When the layer under analysis is a convolutional layer, it is a 4-dimensional matrix we are dealing with. As stated above, the *y* slice of the matrix is the incoming filters, and therefore it is here we use p_{in}. This set represents the filters we want to retain, so we remove any filters not in this set (by using their index values to locate them). We transpose the 4-D matrix into a 2-D representation, as stated in Algorithm 7. The important filters to retain in that layer is calculated by using Algorithm 8, which is explained below. In a similar way to how we retained the important filters in p_{in}, we do the same for p_{out} in the z slice of the matrix. When analysing the fully-connected layers, we follow the same procedure, but we are dealing with a 2-dimensional matrix instead of the 4-dimensional one. As can be seen in Algorithm 7, we repeat for both the medium and fine pruning, continually checking to ensure the β threshold was not breached, until the algorithm has completed and the network is considered fully compressed.

We now explain Algorithm 8. The algorithm accepts a 2 dimensional array, M, where the rows represent the samples to be clustered. We iteratively cluster the samples, starting with k equal to 25% of the number of samples, until we reach k equal to the total number of samples (e.g. with 100 samples, the starting point would be 25 and the end point would be 100). An automated version of the elbow rule [KM13] is used to find the optimal number of clusters, which is also considered the optimal number of filters/nodes in that layer. The *k-means* algorithm clusters the samples into the k value of clusters such that the total within-cluster sum of squares (WCSS) is minimized. By calculating the first order differences between the WCSS, and then the second order differences, we compute sigma σ by getting the differences between them. We find the optimal number of clusters C by finding the maximum positive value of σ. Then for each cluster in C, we locate the index of the sample closest to the cluster centre, creating the set Y, which the algorithm returns.

6.5 Results

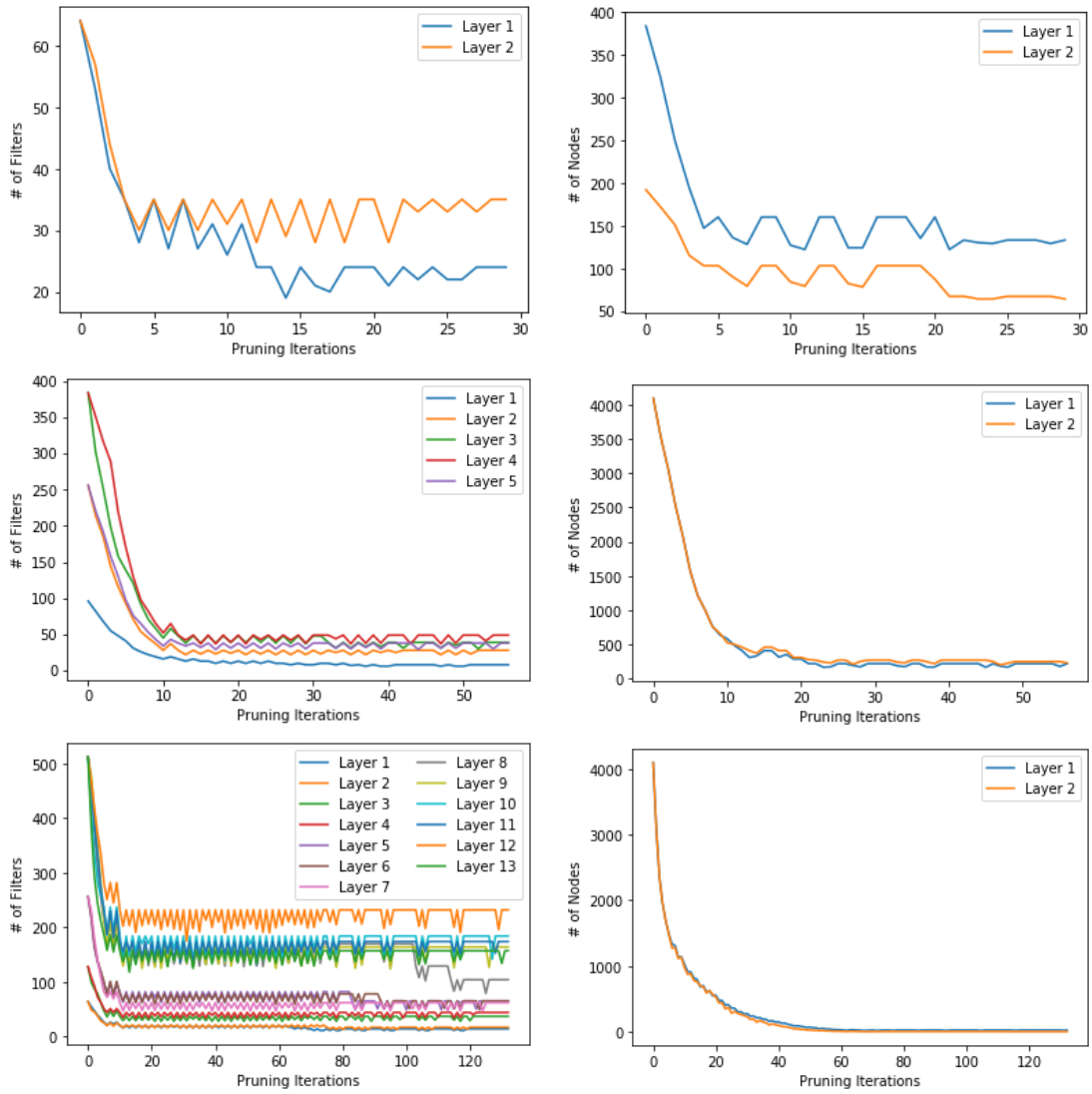

Figure 6.1: Evolution iterations of CifarNet (top row), AlexNet (middle row) and VGG16 (bottom row) networks, while being pruned by Unsupervised PulseNet. The graphs on the left are from the convolutional layers, while the ones on the right are from the fully-connected layers. These results are from the analysis of the CIFAR10 dataset.

To compare inference efficiency between networks, we use the common metric of the number of FLOPs needed to classify a test image. We use the difference between the number of FLOPs for the original network and the Unsupervised PulseNet-pruned network to measure computational improvement. Although we might expect to see similar speedups in inference time and similar reduced energy consumption, in fact we do not. This is due to other layers within the network being non-tensor (batch normalization and pooling layers), along with

the overhead of reading time from hard drive to GPU, which has more of an impact. To fully understand the improvements our proposed method makes on the tested networks, we shall show both the theoretical improvements and the actual improvements on both a GPU and a CPU, with respect to inference time and consumed energy. The theoretical improvement of a pruned network is the percentage of parameters (which is directly related to the number of FLOPs) removed from the original network; so for example if 50% of the parameters are pruned from a network, the theoretical network improvement is a network twice more efficient than the original network. The confusion matrices are located in the appendix section.

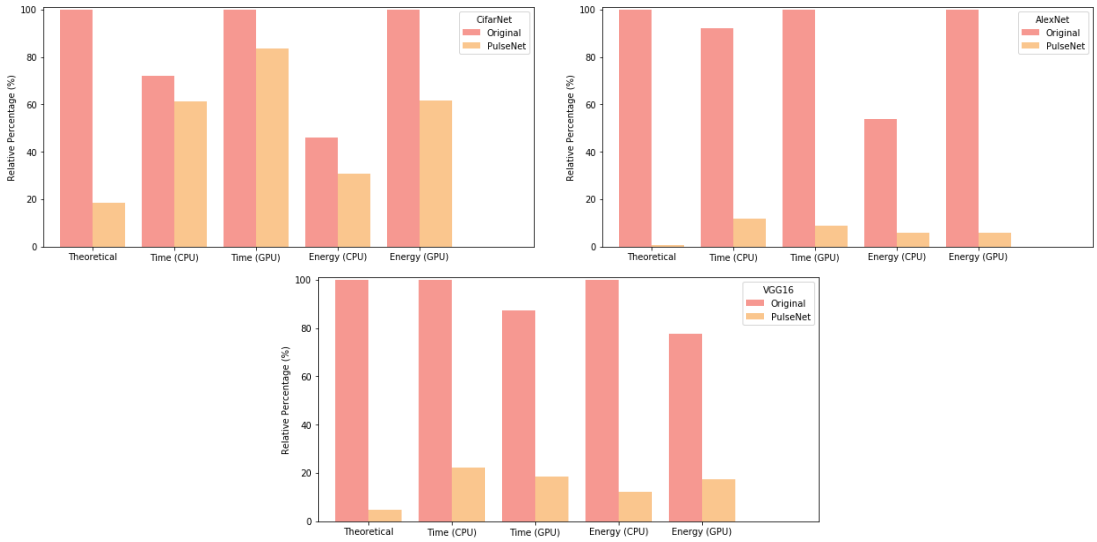

Figure 6.2: Bar-charts for CifarNet (top left), AlexNet (top right) and VGG16 (bottom) networks, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the CIFAR10 dataset.

Table 6.1 shows that Unsupervised PulseNet reduces the complexity of CifarNet, AlexNet and VGG16 by $5.4\times$, $147.1\times$ and $21.1\times$ respectively. This can be compared to similar work on ResNet networks and the CIFAR10 dataset, where [HKD$^+$18, HLW$^+$19] reduced the number of FLOPs by 50% and 52.6% respectively, [DCP17] reduced FLOPs by 34.2%, and [LKD$^+$16] compressed the network by 50%. Although on a different network, the Unsupervised PulseNet reduction in number of FLOPs ranged from 81.39% on an already very small network (CifarNet) to 99.32% on AlexNet, and 95.27% on VGG16.

As can be seen from Figure 6.3, the first few convolutional layers of the networks were pruned the most, followed closely by the last few convolutional layers. This is due to the first part of the convolutional layers being edge, shape

Method	Network Structure	# Parameters	# FLOPs	Accuracy
CifarNet				
Original	$64-64-384-192$	797962 (100%)	1594501 (100%)	85.66
PulseNet	$24-35-133-64$	148593 (18.62%)	296659 (18.61%)	83.20
AlexNet				
Original	$96-256-384-384-256-4096-4096$	46787978 (100%)	93556808 (100%)	90.50
PulseNet	$39-59-132-141-106-340-549$	316571 (0.68%)	631874 (0.68%)	88.79
VGG16				
Original	$64-64-128-128-256-256-256-512-512-512-512-512-512-4096-4096$	33638218 (100%)	67251600 (100%)	91.85
PulseNet	$14-17-37-44-65-65-62-104-164-184-174-232-157-25-6$	1590755 (4.73%)	3178806 (4.73%)	89.32

Table 6.1: Overall accuracies and standard deviations (%) of different CNN architectures, both original and compressed using proposed method Unsupervised PulseNet on the CIFAR10 data set. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

and colour detectors, possible with much overlapping of filters, while the last section of the convolutional part of the network is more class-specific, so the required number of filters in these layers is likely to be proportional to the number of target classes in the dataset. We infer this by examining the Figures 6.6 and 6.9, which have 100 and 200 classes in their respective datasets, and retain more filters within the convolutional layers of their networks. The fully-connected layers of both AlexNet and VGG16 are heavily pruned, which has been shown to be beneficial. In related work, it has been shown that overfitting can occur when a network's fully-connected layers are too dense [LKD+16]. CifarNet's dense layers are less pruned because it is already a narrow network.

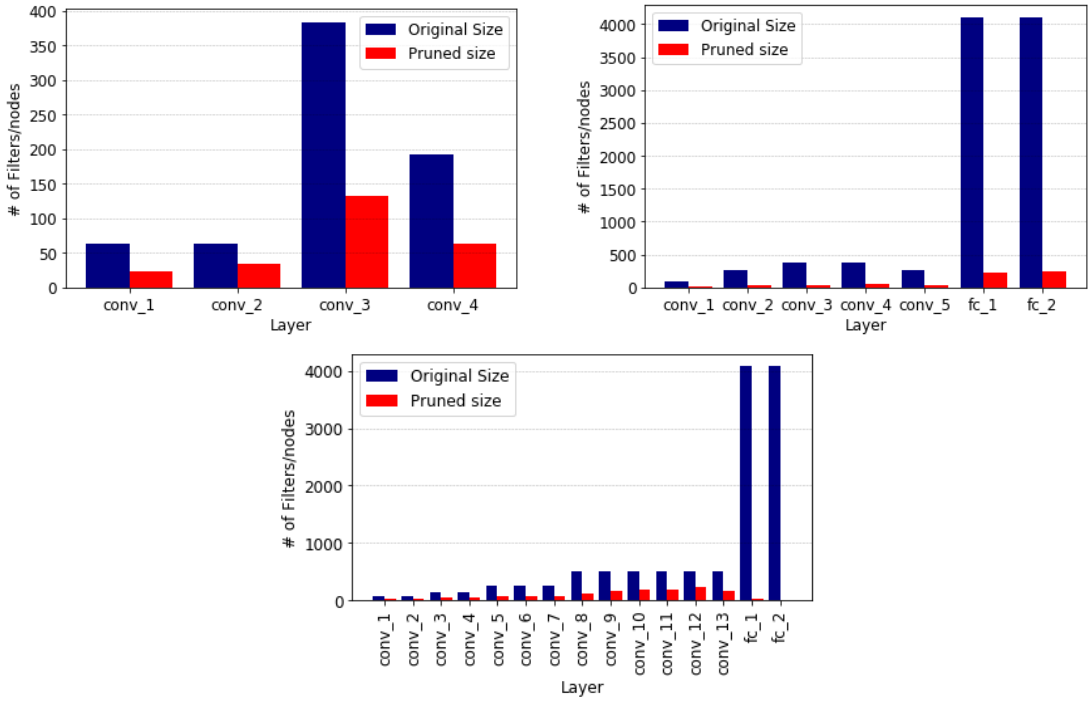

Figure 6.3: Comparison of layers between original and pruned version of CifarNet (top left), AlexNet (top right) and VGG16 (bottom) on the CIFAR10 dataset.

To show the difference between theoretical improvements and actual performances on both a GPU and CPU, we graph the relative percentage differences between them in Figure 6.2. Both AlexNet and VGG16 exhibit near-theoretical improvements, with the performance between the GPU and CPU being very similar, while on CifarNet only the energy consumed on the CPU has near-theoretical improvement. The compressed Unsupervised PulseNet version of CifarNet still benefits from being pruned, though it is a small network, with results beating the original network in efficiency (see Figure 6.2).

The confusion matrices of the original and compressed versions of the CifarNet model on the CIFAR10 dataset are shown in Figure C.1. The original version has an accuracy score of 85.66%, a precision score of 85.59%, recall score of 85.62% and an F1-score of 85.48%, while the Unsupervised PulseNet version has an accuracy score of 83.20%, a precision score of 83.31%, recall score of 83.20% and a F1-score of 82.99%. The original AlexNet, on CIFAR10 has an accuracy score of 90.50%, a precision score of 90.56%, recall score of 90.46% and a F1-score of 90.47%, seen in Figure C.2, and the pruned model has an accuracy score of 88.79%, a precision score of 88.82%, recall score of 88.75% and a F1-score of 88.71%. Finally, the confusion matrices of the VGG16 network are shown in Figure C.3, with the original network having an accuracy score of 91.85%, a precision score of 91.90%, recall score of 91.85% and a F1-score of 91.80%, and the compressed version has an accuracy score of 89.32%, a precision score of 89.59%, recall score of 89.32% and a F1-score of 89.29%.

Figure 6.1 illustrates how Unsupervised PulseNet prunes each layer of the targeted network by iterative pruning and fine tuning. Unsupervised PulseNet initially prunes at a high rate, then compresses and decompresses while reducing its pruning rate, all in an unsupervised manner. We can see that AlexNet and VGG16 have a much smoother approach to their fully compressed states, compared to CifarNet. In AlexNet the convolutional layers converge to a reasonably similar number after the tenth iteration, and gradually removes fewer filters to achieve final compressed state. On the other hand, VGG16 convolutional layers approximately maintain the relative difference in number of filters per layer as it is pruned. There is much more compression and de-compression in VGG16 compared to the other two networks, showing that it was not an easy network in which to find the optimal number of filters to prune within each layer. The dense layers of all three networks maintain roughly the same relative difference compared to their original state, and all are pruned significantly early on in the process, reducing as the layers are thinned out.

[DCP17] used a layer-wise pruning method on a 3-convolutional- and 2-fully-connected-layer network based on AlexNet, applied to the CIFAR10 dataset. They were able to compress it to 9% of its original size while maintaining an accuracy of 80.64%. We reduced our CifarNet to 18.62% of its original size with a classification accuracy of 83.20%. Although they achieved a higher compression ratio, it should be remembered that their network was based on AlexNet with 3 convolutional layers of sizes 96, 256, 256 with the same fully-connected layers as ours. Hence our network has far fewer parameters than theirs, with less

pruning potential. If we compare our version of AlexNet on CIFAR10 we pruned to a ratio of 0.68%.

[RBK+14] proposed a teacher-student network idea called FitNets, that compressed their model down to ~ 2.5M parameters with an accuracy of 91.61% on the CIFAR10 dataset. Comparing our Unsupervised PulseNet version of AlexNet, we achieved a slightly worse classification accuracy of 88.79% but with only ~ 0.6M parameters: a quarter less than FitNets.

In the work of [IHM+16] on SqueezeNet they compressed a network by $50\times$ while maintaining an AlexNet accuracy. Our Unsupervised PulseNet model of AlexNet was compressed by $147.1\times$ with negligible loss in accuracy.

The pruning method of [ZRZ+18] compressed the convolutional layers of VGG16 using the CIFAR10 dataset. The first 4 layers of their version of VGG16 had 30, 46, 94 and 102 filters, with the remaining convolutional layers containing 206 filters, while as can be seen in Figure 6.1 all but one of our version of VGG16 layers were pruned much more. On the first 4 layers, our network had less than half the number of filters compared to theirs, with the rest of the convolutional layers ranges from $1.1\times$ to $3.3\times$ fewer filters (one of our layers retained 26 more filters), making the Unsupervised PulseNet network computationally more efficient.

[HZYN18] pruned the filters in the convolutional layers of VGG16, and on the CIFAR10 dataset reduced FLOPs by 80.6%, while our method reduced FLOPs by 95.27%. This resulted in a speedup of their network by 63.4%, whereas our VGG16 network improved inference time by 78.8%. They had a loss in accuracy of 3.4% compared to Unsupervised PulseNet's loss of 2.5%.

[DDHT18] used an iterative approach called Auto-Balanced Filter Pruning to compress the VGG16 network by 81.4% on the CIFAR10 dataset, whereas our proposed method pruned VGG16 by 95.3%. Unsupervised PulseNet lost an average of 2% accuracy while [DDHT18] lost approximately 0.5%. [LAT18] compressed AlexNet and VGG16 on the CIFAR10 dataset by $10\times$ and VGG16 by $19.6\times$, while our method reducing AlexNet by $147.1\times$ and VGG16 by $21.1\times$. [ZNZ+19] pruned VGG16 by $3.8\times$, with a 2% loss in accuracy, whereas we pruned it by $21.1\times$ with approximately the same loss in accuracy. [BGP19] used an iterative pruning approach, that simulated the pruning of filters/nodes by using binary matrices. On the CIFAR10 dataset this compressed the CifarNet, AlexNet and VGG16 networks by $4.2\times$, $22.9\times$, and $8.2\times$, losing 3.6%, 2%, 0.9%

accuracy, respectively. In comparison, Unsupervised PulseNet pruned CifarNet by $5.4\times$, AlexNet by $147.1\times$ and VGG16 by $21.1\times$, with a classification loss of 2.46%, 1.7% and 2.5% respectively.

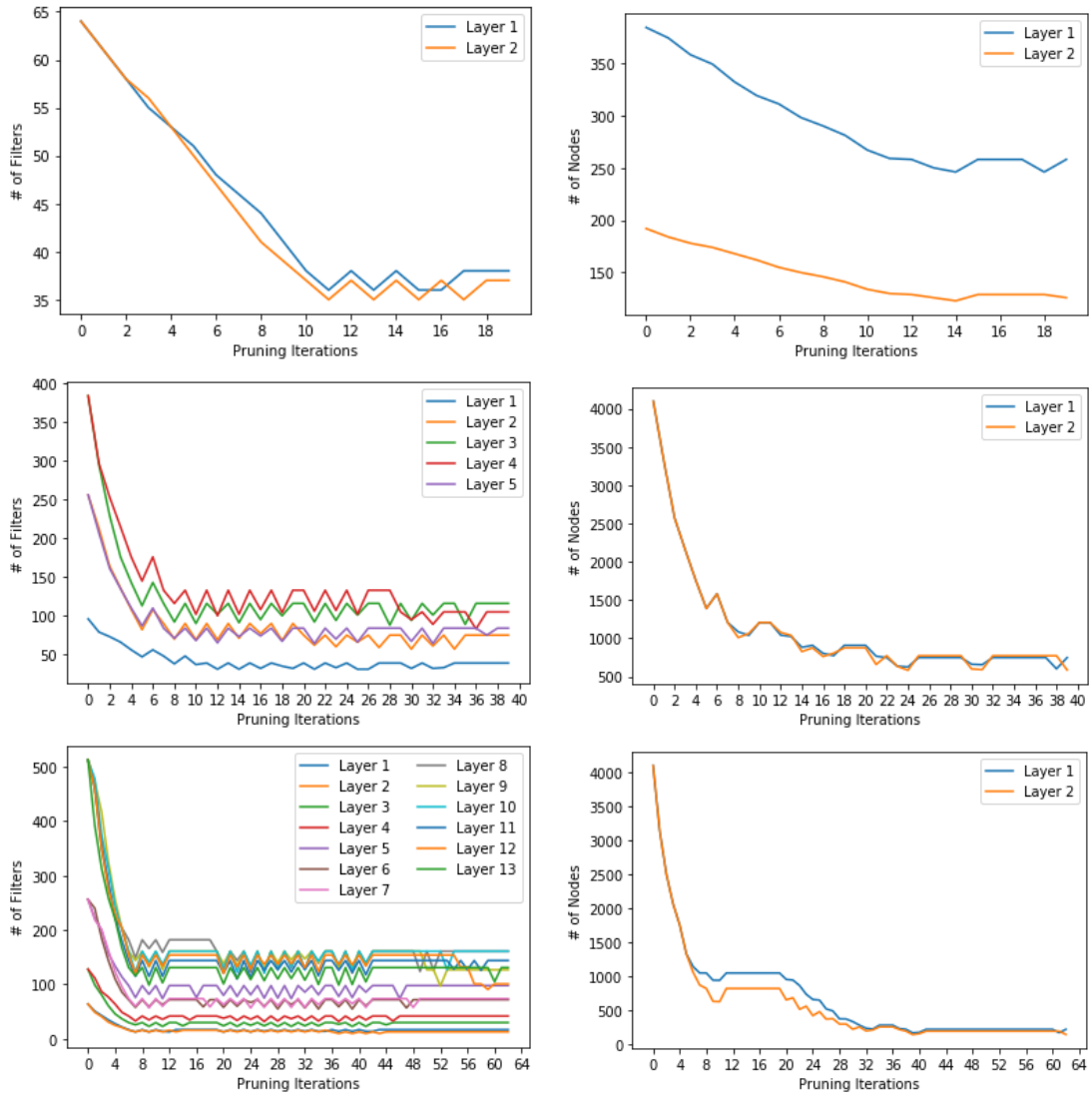

Figure 6.4: Evolution iterations of CifarNet (top row), AlexNet (middle row) and VGG16 (bottom row) networks, while being pruned by Unsupervised PulseNet. The graphs on the left are from the convolutional layers, while the ones on the right are from the fully-connected layers. These results are from the analysis of the CIFAR100 dataset.

Unsupervised PulseNet reduced the complexity of CifarNet by $2.5\times$, AlexNet by $22.4\times$ and VGG16 by $28.4\times$, as shown in Figure 6.2. The reduction in the number of FLOPs ranged from 60.27% on CifarNet to 96.48% on VGG16, with 95.53% on AlexNet. The pruned layers follow a similar pattern to the CIFAR10 dataset, with the middle layers being less pruned compared to the first and last few convolutional layers, as shown in Figure 6.6. The fully-connected layers

Method	Network Structure	# Parameters	# FLOPs	Accuracy
CifarNet				
Original	$64-64-384-192$	815332 (100%)	1629061 (100%)	58.71
PulseNet	$38-37-258-129$	323394 (39.66%)	645669 (39.63%)	56.82
AlexNet				
Original	$96-256-384-384-256-4096-4096$	47156708 (100%)	94294088 (100%)	70.56
PulseNet	$39-75-116-105-84-748-773$	2582926 (5.48%)	5161780 (4.47%)	68.40
VGG16				
Original	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	34006948 (100%)	67988880 (100%)	67.71
PulseNet	$17-13-30-42-98-72-74-161-127-161-144-101-131-228-201$	1197197 (3.52%)	2391010 (3.52%)	65.38

Table 6.2: Overall accuracies and standard deviations (%) of different CNN architectures, both original and compressed using Unsupervised PulseNet on the CIFAR100 data set. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

are less pruned than on CIFAR10, possible due to CIFAR100 having $10\times$ the number of classes.

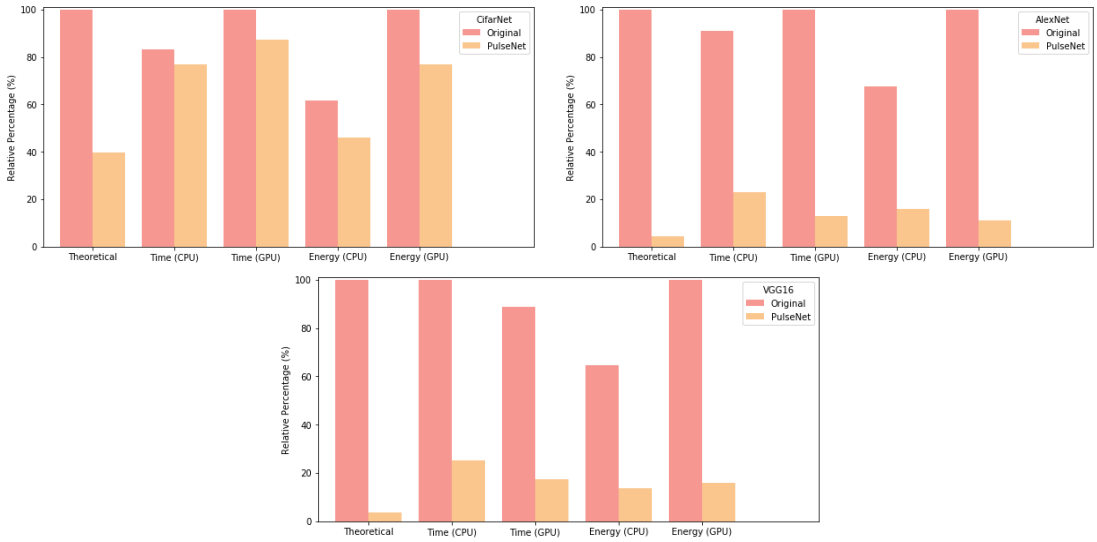

Figure 6.5: Bar-charts for CifarNet (top left), AlexNet (top right) and VGG16 (bottom) networks, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the CIFAR100 dataset.

We compare the efficiency between the original and compressed networks with respect to the theoretical improvement, and actual results on a GPU and CPU, in Figure 6.5. Confirming the results of the CIFAR10 dataset, both AlexNet and VGG16 achieve close to the expected theoretical increase in efficiency, while the specifically designed CifarNet model improves in efficiency with the Unsupervised PulseNet version over the original version but does not reach the theoretical improvement. The original version of CifarNet had an accuracy score of 58.71%, a precision score of 59.28%, recall score of 58.71% and a F1-score of 58.35%, as shown in Figure C.4, compared with the Unsupervised PulseNet compressed version with an accuracy score of 56.82%, a precision score of 57.65%, recall score of 56.82% and a F1-score of 56.45%.

The confusion matrices of both AlexNet versions are shown in Figure C.5. The full version had an accuracy score of 70.56%, a precision score of 71.33%, recall score of 70.56% and a F1-score of 70.51%, while the pruned version had an accuracy score of 56.82%, a precision score of 57.65%, recall score of 56.82% and a F1-score of 56.45%.

Lastly, the results of the VGG16 networks are shown in Figure C.6. We see that the original network had an accuracy score of 67.71%, a precision score

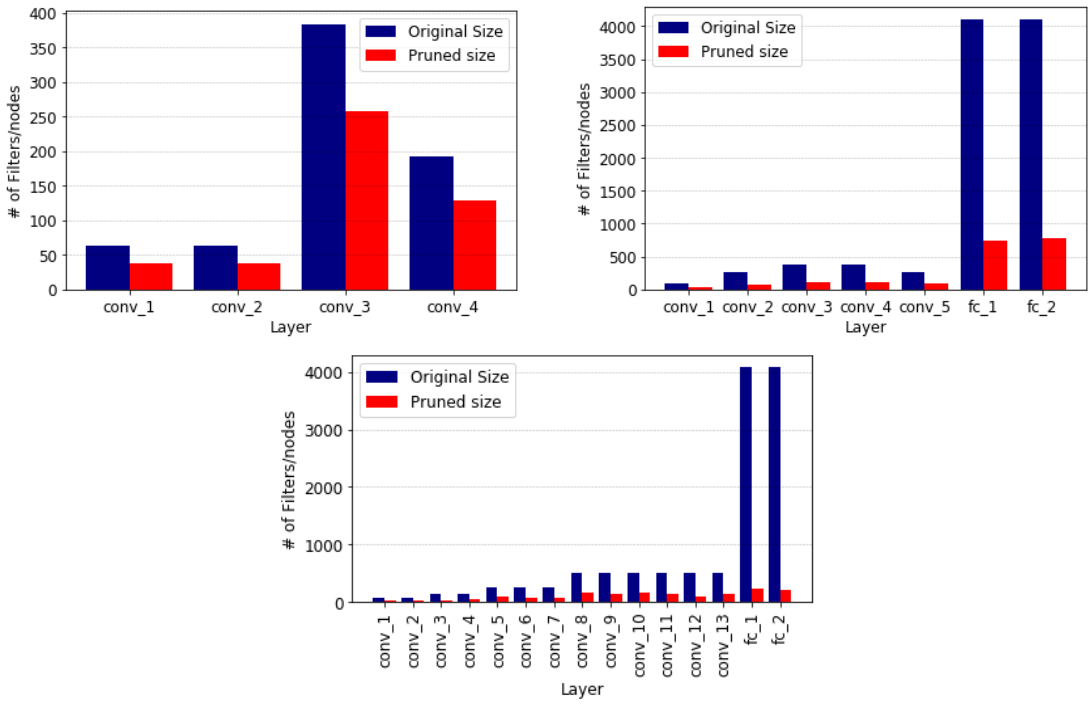

Figure 6.6: Comparison of layers between original and pruned version of CifarNet (top left), AlexNet (top right) and VGG16 (bottom) on the CIFAR100 dataset.

of 70.01%, recall score of 67.71% and a F1-score of 68.18%, while the pruned network had an accuracy score of 65.38%, a precision score of 66.64%, recall score of 65.38% and a F1-score of 65.48%.

Next we analyse the graphs of the pruning iterations in Figure 6.4. Starting with the dense fully-connected layers, it can be seen that the compression process was more disruptive compared to that of the dense layers on the CIFAR10 dataset. We can see, especially in the graphs for AlexNet and VGG16, that Unsupervised PulseNet had to revert back to the best previous state several times, being unable to regain accuracy loss in validation. It is clear on the VGG16 graph, between the 20^{th} and 30^{th} pruning iterations, that Unsupervised PulseNet gradually reduced both layers from about 800–100 nodes to approximately 200 nodes in a step-like decay: the "pulse" aspect of our method (the compression and decompression of the network).

The convolutional layers of the three networks are initially pruned heavily, but then go through an exploratory phase to find clusters to guide redundant filter removal.

[RBK+14] proposed the FitNets method, which compressed their model down

to ~ 2.5M parameters with an accuracy of 64.96% on the CIFAR100 dataset. Comparing our Unsupervised PulseNet version of VGG16, we maintained a classification accuracy of 65.38% with ~ 2.4M parameters, improving both accuracy and compression. [ZNZ+19] pruned VGG16 by $1.6\times$ without loss in accuracy, while Unsupervised PulseNet compressed VGG16 by $28.4\times$ with just over 2% classification loss. [BGP19] on the CIFAR100 dataset compressed the CifarNet, AlexNet and VGG16 networks by $2.9\times$, $7.1\times$, and $8.1\times$, losing 4.5%, 2.3%, 1.1% accuracy, respectively. In contrast, Unsupervised PulseNet pruned CifarNet by $2.5\times$, AlexNet by $22.4\times$ and VGG16 by $28.4\times$, with accuracy losses of 1.9%, 2.16% and 2.3%, respectively.

The last dataset for analysis is Tiny-ImageNet. Unsupervised PulseNet reduces the network complexity of PulseNet's CifarNet by $2.4\times$, eliminating 57.65% of the number of FLOPs. It decreases the complexity of AlexNet by $37.3\times$ which eliminates 97.32% of the number of FLOPs. And it decreases the VGG16 complexity by $19.1\times$, eliminating 95.75% of FLOPs. The pruning of the layers, displayed in Figure 6.9, shows that on CifarNet the dense layers are pruned less than on the other datasets, due to the Tiny-ImageNet number of classes.

All three networks on the three datasets show a very similar pattern: the first and last convolutional layers are pruned most because of overlapping of edge, shape and colour detectors that occur in the first few convolution layers, and the last few convolution layers are pruned less because they contain more information about the target classes. All the dense fully-connected layers are significantly reduced, reconfirming that these layers contain many redundant nodes that can cause over-fitting.

From the efficiency graphs in Figure 6.8 we see that the CifarNet network struggles to meet the theoretical improvement, but the Unsupervised PulseNet pruned version of the model still beats the original version in all the experiments. However, both the other two networks come very close to the theoretically possible improvement in performance on the compressed models, easily achieving a more computationally efficient network compared to the original networks on the GPU and CPU results.

The confusion matrices of the CifarNet models are shown in Figure C.7. The original version has an accuracy score of 41.08%, precision score of 47.96%, recall score of 41.08% and F1-score of 41.71%, while the Unsupervised PulseNet version has an accuracy score of 39.70%, precision score of 46.35%, recall score of 39.70% and F1-score of 40.41%.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
CifarNet				
Original	$64-64-384-192$	4373576 (100%)	8745349 (100%)	41.08
PulseNet	$47-49-210-126$	1852402 (42.35%)	3703545 (42.35%)	39.70
AlexNet				
Original	$96-256-384-384-256-4096-4096$	59100744 (100%)	118181960 (100%)	52.47
PulseNet	$29-134-170-167-108-198-214$	1586747 (2.68%)	3171062 (2.68%)	50.13
VGG16				
Original	$64-64-128-128-256-256-256-512-512-512-512-512-512-4096-4096$	40708104 (100%)	81390992 (100%)	55.29
PulseNet	$17-17-40-32-82-84-109-149-176-188-244-215-186-25-14$	2137115 (5.25%)	4270690 (5.25%)	52.84

Table 6.3: Overall accuracies and standard deviations (%) of different CNN architectures, both original and compressed using Unsupervised PulseNet on the Tiny-Imagenet data set. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

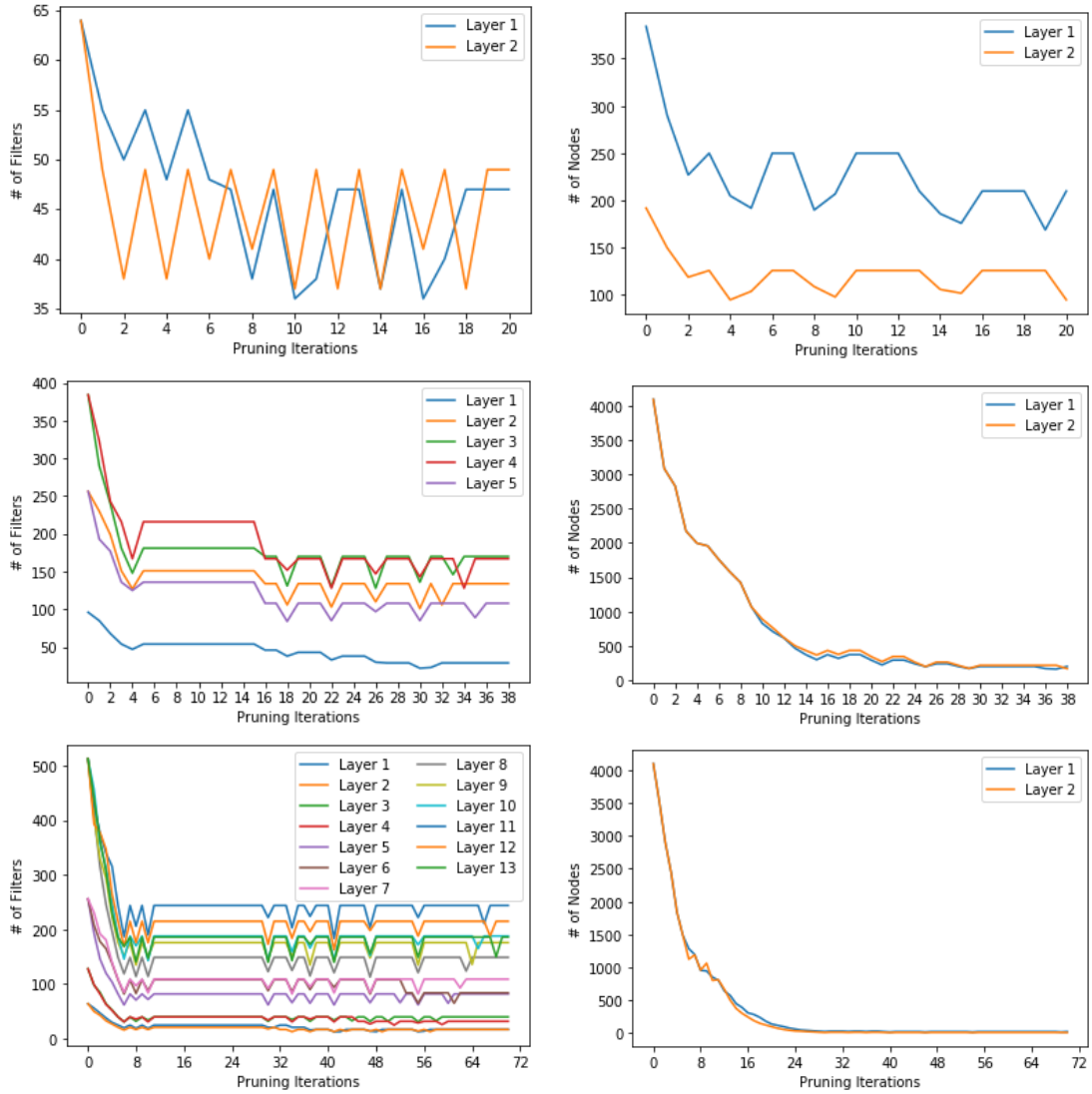

Figure 6.7: Evolution iterations of CifarNet (top row), AlexNet (middle row) and VGG16 (bottom row) networks, while being pruned by Unsupervised PulseNet. The graphs on the left are from the convolutional layers, while the ones on the right are from the fully-connected layers. These results are from the analysis of the Tiny-ImageNet dataset.

Figure C.8 shows both confusion matrices of the AlexNet models, with the original network achieving an accuracy score of 52.47%, a precision score of 58.12%, recall score of 52.47% and a F1-score of 52.96%, and the compressed version scoring an accuracy score of 50.13%, a precision score of 55.86%, recall score of 50.13% and a F1-score of 50.75%.

The confusion matrices in Figure C.9 refer to the VGG16 pruned and original networks. The original network has an accuracy score of 55.29%, a precision score of 60.23%, recall score of 55.29% and a F1-score of 55.66%. The Unsu-

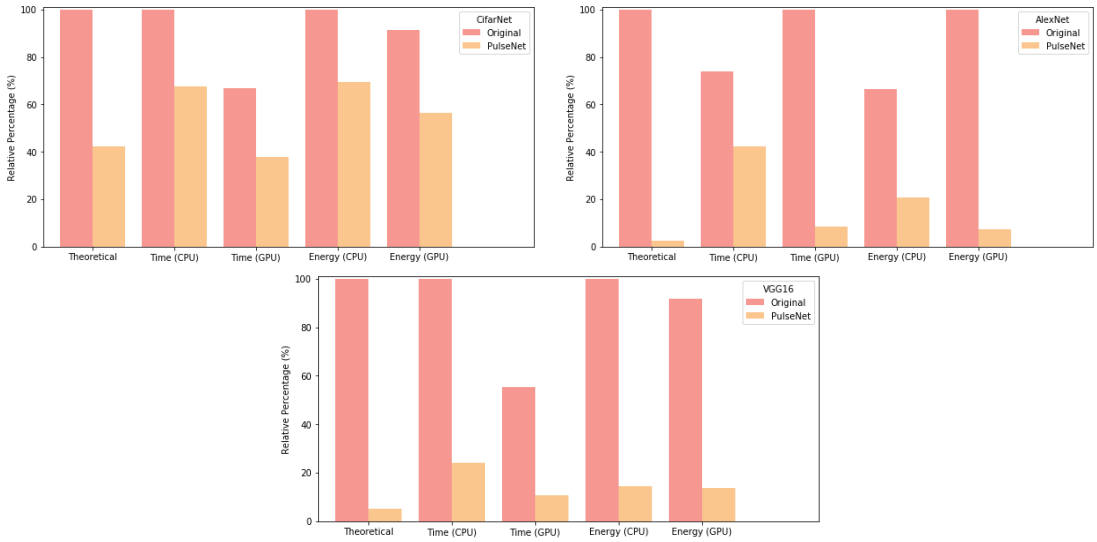

Figure 6.8: Bar-charts for CifarNet (top left), AlexNet (top right) and VGG16 (bottom) networks, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the Tiny-ImageNet dataset.

pervised PulseNet network has an accuracy score of 52.84%, precision score of 61.83%, recall score of 52.84% and F1-score of 54.48%.

The graphs of the pruning iterations from Figure 6.7 show that the dense layer patterns are more similar to those of the CIFAR10 dataset rather than the CIFAR100 dataset. CifarNet's convolutional layers were quite difficult to prune, as it was already a small network. However, Unsupervised PulseNet is still able to reduce the number of filters in both layers of this already constrictive network. Unlike the CifarNet model, the convolutional layers of the other two datasets are near full compression by the 10^{th} pruning iteration, but our method still searches the space to find a *good* minimum number of filters per layer, by only retaining important filters and nodes that improve generalization.

[DCP17] compressed both AlexNet and VGG16 on the ImageNet dataset and, although we cannot perform a direct comparison because Tiny-ImageNet is a only subset of that dataset, it is still interesting to compare the results. They were able to compress AlexNet by $17.5\times$ and VGG16 by $13.3\times$, compared to Unsupervised PulseNet which pruned AlexNet and VGG16 by $37.3\times$ and $19.1\times$, respectively, showing we can achieve better compression rates. The pruning method of [LWL17] called ThiNet, applied to the ImageNet dataset, reduced VGG16 down to 8.34M parameters, while our compressed version of VGG16 on Tiny-ImageNet is almost $4\times$ less with 2.1M parameters. [YLC+18] used a neuron

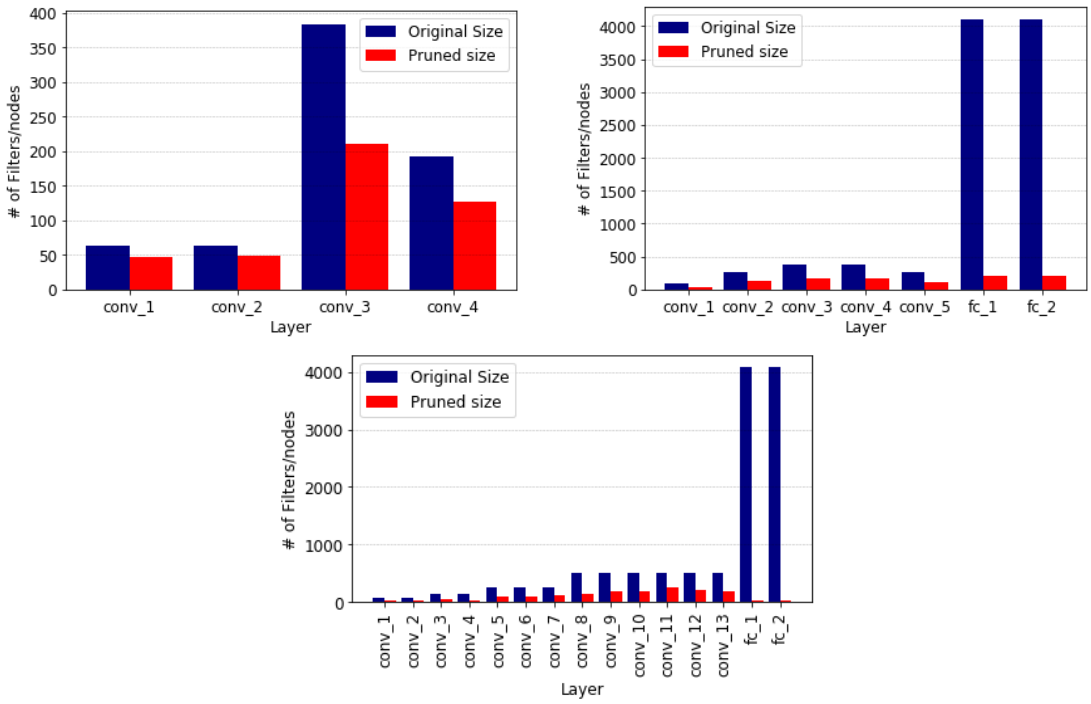

Figure 6.9: Comparison of layers between original and pruned version of CifarNet (top left), AlexNet (top right) and VGG16 (bottom) on the Tiny-ImageNet dataset.

importance score propagation (NISP) approach to prune AlexNet on the ImageNet dataset. Their NISP AlexNet had a reduction in the number of FLOPs by 67.85%, compared to Unsupervised PulseNet AlexNet on Tiny-ImageNet with a reduction in FLOPs of 97.32%. [KPY+15] compressed both AlexNet and VGG16 networks on the ImageNet dataset, with AlexNet being pruned by $5.5\times$ and VGG16 by $7.4\times$ with 1.7% and 0.55% loss in accuracy respectively. In contrast, we pruned AlexNet by $37.3\times$ with a accuracy loss of 2.3%, and VGG16 by $19\times$ with a loss of 2.5%. The method of [LAT18], called Single-Shot Network Pruning, compressed AlexNet to 10.1% and VGG16 to 20% of their original sizes, while *unsupervised PulseNet* compressed AlexNet and VGG16 to 2.7% and 5.2% of their original sizes, respectively, on the Tiny-ImageNet dataset within acceptable accuracy loss. Comparing to the results of [ZZLS18] on AlexNet, where they achieved a theoretical improvement of $13\times$, Unsupervised PulseNet improved its version of the network by over $3\times$ their compression, to $37.1\times$ smaller. The method of [BGP19] on the Tiny-ImageNet dataset compressed the CifarNet, AlexNet and VGG16 networks by $3.9\times$, $5.2\times$, and $6.2\times$, losing accuracy 2.91%, 3.75%, 2%, respectively. In contrast, Unsupervised PulseNet pruned CifarNet by $2.4\times$, AlexNet by $37.3\times$ and VGG16 by $19.1\times$, with a classification loss of 1.4%, 2.3% and 2.45%, respectively.

			Original		Unsupervised PulseNet		
Dataset	CNN	Processor	Storage (MB)	Energy per image (mJ)	Time per image (ms)	Storage (MB)	Energy (mJ)	Time per image (ms)
Cifar10	CifarNet	CPU	3.044	0.06	2.20	0.567	0.04	1.87
-	-	GPU	-	0.13	3.05	-	0.08	2.55
-	AlexNet	CPU	178.482	1.11	32.05	1.208	0.12	4.11
-	-	GPU	-	2.06	34.82	-	0.12	3.06
-	VGG16	CPU	128.32	0.98	27.68	6.068	0.12	6.12
-	-	GPU	-	0.71	24.18	-	0.17	5.13
Cifar100	CifarNet	CPU	3.11	0.08	2.43	1.234	0.06	2.24
-	-	GPU	-	0.13	2.92	-	0.10	2.55
-	AlexNet	CPU	179.889	1.14	32.01	9.853	0.27	8.14
-	-	GPU	-	1.69	35.22	-	0.19	4.59
-	VGG16	CPU	129.726	0.98	27.66	4.567	0.21	6.97
-	-	GPU	-	1.52	24.57	-	0.24	4.76
Tiny ImageNet	CifarNet	CPU	16.684	0.23	8.05	7.066	0.16	5.45
-	-	GPU	-	0.21	5.38	-	0.13	3.06
-	AlexNet	CPU	225.451	1.74	32.05	6.053	0.54	18.28
-	-	GPU	-	2.62	43.40	-	0.19	3.75
-	VGG16	CPU	155.289	1.92	50.69	8.152	0.28	12.21
-	-	GPU	-	1.76	28.05	-	0.26	5.38

Table 6.4: Computational results on datasets CIFAR10, CIFAR100 and Tiny-ImageNet using three CNNs; CifarNet, AlexNet and VGG16. It shows the storage space each network requires, the inference speed per image and the energy consumed per image of both the original and compressed networks using a batch size of a single image.

Table 6.5: Overall accuracies (%) of different CNN architectures, both original and compressed using a single pruning iteration of Unsupervised PulseNet on the Tiny-Imagenet data set. The entries in red show the percentage size of the networks.

Method	# Parameters	Accuracy
CifarNet		
Original	4373576 (100%)	41.08
PulseNetOne	755991 (14.99%)	32.48
AlexNet		
Original	59100744 (100%)	52.47
PulseNetOne	4326174 (7.32%)	47.66
VGG16		
Original	40708104 (100%)	55.29
PulseNetOne	9737378 (23.92%)	54.80

Table 6.6: Overall accuracies (%) of different CNN architectures, both original and compressed using a single pruning iteration of Unsupervised PulseNet on the CIFAR10 data set. The entries in red show the percentage size of the networks.

Method	# Parameters	Accuracy
CifarNet		
Original	797962 (100%)	85.66
PulseNetOne	84903 (10.64%)	79.42
AlexNet		
Original	46787978 (100%)	90.50
PulseNetOne	7130488 (15.24%)	91.02
VGG16		
Original	33638218 (100%)	91.85
PulseNetOne	5419117 (16.11%)	90.98

6.5.1 Further Experiments

We experimented with only using a single pruning iteration in the aim to significantly reduce the time it required to extract a small compressed network. There were a few issues with this idea; we had less control over the loss in accuracy, as we can see from the results of CifarNet network in all 3 of the tables; 6.6, 6.7

Table 6.7: Overall accuracies (%) of different CNN architectures, both original and compressed using a single pruning iteration of Unsupervised PulseNet on the CIFAR100 data set. The entries in red show the percentage size of the networks.

Method	# Parameters	Accuracy
CifarNet		
Original	815332 (100%)	58.71
PulseNetOne	230168 (28.23%)	52.72
AlexNet		
Original	47156708 (100%)	70.56
PulseNetOne	8422188 (17.86%)	69.18
VGG16		
Original	34006948 (100%)	67.71
PulseNetOne	7362504 (21.65%)	66.16

and 6.5, where we lost between 6% and 8%. Another lesser result was that we did not achieve as high of a compression with respect to the iteration process of unsupervised PulseNet, as can be seen in all 3 tables, and especially noticeable using the AlexNet network on CIFAR10 dataset. We pruned the network down to 15.24% of its original size when using the single pruning iteration PulseNet, while on the version this chapter is mainly focused on, we were able to prune the same network on the same dataset down to 0.68% of its original size. Although, when talking about the same 2 experiments, the single shot approach had a small gain in accuracy of 0.5%, while the unsupervised PulseNet from this chapter had a decrease of 1.7%, still within the threshold of predefined accuracy loss. All of the results of this section of the chapter has given reason for even further investigation, and experiments tweaking the original Unsupervised PulseNet to develop a new variation.

6.6 Conclusion

We proposed an effective neural network pruning algorithm based on finding clusters of filters and nodes in each layer via unsupervised K-means clustering. By retaining only the filters and nodes close to cluster centroids, we achieve state-of-the-art compression with negligible loss in accuracy. Our approach does not require any additional software or hardware, is automated, and can be

implemented on any type of CNN. We demonstrated our method, which we call Unsupervised PulseNet, on three benchmark datasets using well-established neural networks in the area of computer vision.

We have shown our unsupervised pruning method is a powerful compression algorithm for CNNs, but one of the main pitfalls is the iterative way it prunes networks. In a way Unsupervised PulseNet can be thought as a theoretical result of pruning networks, while we would like to demonstrate a more industrial practical version of our method. This is to show our proposed method could be implemented in a fast version, more suitable to real world applications. We did this in Section 6.5.1 of this chapter, but explore it in greater depth and evaluation in the next chapter, Chapter 7, with a fast version of unsupervised PulseNet.

Chapter 7

PulseNetOne: Fast Unsupervised Compression of Convolutional Neural Networks for Remote Sensing and Scene classification

7.1 Motivation

Following the impressive results from Chapter 6, we decided to tweak unsupervised PulseNet (see Section 6.4.3) in order to improve the training time. We take all the benefits of Unsupervised PulseNet from the previous chapter, and by altering the method to be a lot faster during the training phase, we demonstrate how it can be a very practical tool in industry evaluating it on benchmark remote sensing and scene data. We chose this type of data to evaluate the new version of the algorithm, for 2 reasons; we see great benefit in the area for a highly efficient pruned network (installation on Satellites, network inference on mobile monitoring IoT devices), and also because experimental results in the area would help to advance image recognition tasks showing the potential advantages of pruning well established CNNS already deployed in the area of remote sensing and scene classification. Scene classification is an important aspect of image/video understanding and segmentation. However, remote sensing scene classification is a challenging image recognition task, partly due to the limited training data, which causes deep learning Convolutional Neural Networks (CNNs) to over-fit. Another difficulty is that images often have very

different scales and orientation. Yet another is that the resulting networks may be very large, again making them prone to over-fitting and also unsuitable for deployment on memory- and energy-limited devices. We propose an efficient deep learning approach to tackle these problems. We use transfer learning to compensate for the lack of data, and data augmentation to tackle varying scale and orientation. To reduce network size we use a novel unsupervised learning approach based on k-means clustering, applied to all parts of the network: most network reduction methods use computationally expensive supervised learning methods, and apply only to the convolutional or fully-connected layers but not both. In experiments we set new standards in classification accuracy on six remote sensing image datasets.

7.2 Introduction

Remote sensing image classification has gained in popularity as a research area, due to the increase in availability of satellite imagery and advancements in deep learning methods for image classification. A typical image contains objects and natural scenery, and the algorithms must understand which parts of the image are important and relate to the class, and which parts can be considered noise. Scene classification has been applied to various industrial products such as drones and autonomous robots, to improve their predictability at understanding scenes.

Much early work concentrated on using hand-crafted methods such as colour histograms, texture features, scale-invariant feature transform (SIFT) or histograms of oriented gradient (HOG). Colour histograms were very simple to implement but, though translation- and rotation-invariant, they were unable to take advantage of spatial information in an image [PNDS15]. For the analysis and classification of aerial and satellite images, texture features were commonly used, including Gabor features, co-occurrence matrices and binary patterns [dSPdST10]. SIFT used the gradient information about important keypoints to describe regions of the image. There are different types of SIFT (sparse SIFT, PCA-SIFT and SURF) which are all highly distinctive and are scale-invariant, illumination-invariant and rotation-invariant [KMW11]. HOG calculates the distribution of the intensities and directions of the gradients of regions in the image, and has had great success at edge detection and identifying shape details in images [CZY+16]. Both SIFT and HOG use features to represent local regions of an image, and therefore reduce their effectiveness by not

taking important spatial information into account. To improve these methods, creating global feature representation *bag of visual word* models were introduced [BCD15]. Advances in these models included using pooling techniques such as spatial pyramid matching [HCLD16].

A number of unsupervised approaches have been explored for remote sensing classification problems. Principal component analysis (PCA) and k-means clustering were successful early methods. More recently, auto-encoders have been used in the area as an unsupervised model, which involve reconstructing an image after forcing it through a bottleneck layer. These unsupervised methods improved on hand-crafted features techniques, but distinct class boundaries were hard to define because the data was unlabelled. For this reason supervised learning was more attractive, especially for convolutional neural networks (CNNs) from the field of deep learning, which have been responsible for state-of-the-art results in image classification [ZZZ16a, ZZZ+16b].

Given the unmatched power of deep learning for image classification, it is natural to investigate the usefulness of CNNs on remote sensing data problems [ZDZ15, ZZD16]. CNN models such as AlexNet [KSH12] and VGG16 [SZ14] have demonstrated their ability to extract relevant informative features that are more discriminative than extracted hand-crafted features. [GWGL14, ZWS+14] used a promising strategy of extracting the CNN activations of the variously scaled local regions, and pooling them together into a bag of local features. Their networks were pre-trained on ImageNet, similar to [WWWY15], demonstrating how using a good initialization of parameters can increase network classification accuracy. Other related works show that by avoiding pooling and instead focusing on multi-scale CNN structures, competitive results can be achieved [HJL16].

This type of image classification is very challenging for several reasons. Firstly, although remote sensing datasets are increasing in size, most are still considered small in deep learning terms. This means that we often have insufficient training data to obtain high classification accuracy. To have a fair comparison between our method and related work in this area, we set up the experiments in the same manner. This meant that the training dataset had a very limited number of samples, adding to the problem difficulty. To tackle this problem we use *transfer learning*, which uses other data to provide a good initialization point for the network parameters. A second problem is that images from the same class can have very different scales and/or orientation. To address

this issue we apply a standard method from deep learning image recognition: *data augmentation*. A third problem is that high-resolution satellite images can contain overlapping classes, which can have an inverse effect on classification accuracy [Che13]. Although the method in [Che13] has had great success, it is very dependent on how the initial low-level hand-crafted features are extracted, which in turn relies greatly on domain knowledge. Using CNNs to extract the relevant features eliminates the need for domain knowledge and hand-crafted features. A fourth problem is that training a CNN on images can lead to very large networks that are prone to over-fitting, and unsuitable for deployment on memory- and energy-limited devices.

We propose a novel unsupervised learning approach, based on k-means clustering, for pruning neural networks of unwanted redundant filters and nodes. We find optimal clusters within the filters/nodes of each layer, and discard those furthest from the centre along with all their associated parameters. Our new method, which we call PulseNetOne, combines these techniques with fine-tuning phases to recover from any loss in accuracy. Extensive experiments with various datasets and neural networks were carried out to illustrate the performance and robustness of our proposed pruning technique. We compare it with other state-of-the-art remote sensing classification algorithms, and experiments show that it significantly outperforms them in classification accuracy and regularization while generating much less complex networks. In the area of remote sensing and scene classification, although both AlexNet and VGG16 are popular CNNS in this area, we believe we are the first to display how pruning can significantly improve results in this image recognition task.

The main contributions of this research can be summarized as follows:

- We introduce a novel approach, called PulseNetOne, that rapidly and automatically selects redundant filters/nodes in a CNN, removes them and compresses the model, all in a single pruning iteration.
- Similar to Unsupervised PulseNet from the previous chapter (see Algorithm 6.4.3), the method uses an unsupervised algorithm (K-means clustering) to automatically cluster similar filters and nodes, but differs as the extraction of the new smaller network is considerably faster.
- It achieves state-of-the-art results, pruning VGG16 and AlexNet under the analysis of benchmark remote sensing and scene image datasets, improving the network's classification acuracy.

The rest of this chapter is organised as follows. Section 7.3 discusses the related work on various methods of remote sensing image classification. The datasets and CNNs used are described in Section 7.4, along with our proposed method. The results are given in Section 7.5 as well as their evaluation and discussions. Finally, Section 7.6 concludes the chapter.

7.3 Related Work

A well-established method that has been very successful for satellite image recognition is the *bag of visual words* (BOVW) approach. This usually involves (i) extracting hand-made features, that is SIFT and HOG; (ii) using a clustering algorithm to group the features, thus creating a BOVW with a defined centre; and (3) forming feature representations using histograms, by mapping the learned features onto the nearest cluster centre [DT05, Low04, CDF+04, BCTD17, LLM14, CZS+16]. Both [LSP06] and [YN08] have employed this technique to remote sensing classification. To obtain more meaningful features, spatial pyramids and randomized spatial partitions were used. [LHSL17] used a CNN, originally trained on the ImageNet dataset, with spatial pyramid pooling, and only fine-tuned the fully-connected layers of the network. The spatial pyramid was inserted between the convolutional and fully-connected layers to automatically learn multi-scale deep features, which are then pooled together into sub-regions.

To remove the need for domain-knowledge hand-crafted features, unsupervised learning was used to learn basis functions to encode the features. By constructing the features using the training images instead of hand-crafted ones, better discriminative features are learned to represent the data. Some of the more popular unsupervised methods implemented in this research area include k-means, PCA and auto-encoders [Che13, SYXS12, ZDZ14].

[CYY+18] claimed that an important part of remote sensing classification was to overcome the problems of within-class diversity and between-class similarity, which are both major challenges. The authors proposed to train a CNN on a new discriminative objective function that imposes a metric learning regularization term on the features. At each training iteration random image samples are used to construct similar and dissimilar pairs, and by applying a constraint between the pairs a hinge loss function is obtained. Unlike our proposed algorithm, which is unsupervised and fast, their method requires a number of parameters

to be selected, which most likely will vary depending on the data, and has a complex training procedure that is relatively inefficient. Their results reinforce the idea that, unlike most other approaches that only use the CNN for feature extraction, better performance is achieved by training all the layers within the network.

[SHK19] designed a type of dual network in which features were extracted from the images based on both the objects and the scene of the image, and fused them together, hence the name FOSNet (fusion of object and scene). The authors trained their network using a novel loss function (scene coherence loss) based on the unique properties of the scene. [ZLCD16] fused both local and global features by first partitioning the images into dense regions, which were clustered using k-means. A spatial pyramid matching method was used to connect local features, while the global features were extracted using multi-scale completed local binary patterns which were applied to both gray scale and Gabor filter feature maps. A filter collaborative representation classification approach is used on both sets of features, local and global, and images are classified depending on the minimal approximation residual after fusion. [HZCZ17] also used a spatial pyramid idea with the AlexNet CNN as its main structure. The authors find that, although AlexNet has shown great success in scene classification and as a feature extractor, because of limited training datasets it is prone to over-fitting. By using a technique of side supervision on the last three convolutional layers, and spatial pyramid pooling before the first fully-connected layer, the authors claim that this unique AlexNet structure, named AlexNet-SPP-SS, helps to counteract over-fitting and improve classification. Our work also shows how both AlexNet and VGG16 are prone to over-fitting due to lack of training data, but by pruning redundant filters/nodes we add a strong regularization to the networks, which prevents over-fitting and increases model efficiency.

[TWK17] argue that using a deeper network (GoogLeNet) and extracting features at three stages helps to improve the robustness of the model, allowing low-, mid- and high-level features to contribute more directly to the classification. Instead of the usual additive approach to pooling features, they show that a product principle works better. Their work includes experiments on Scene15, MIT67 and SUN397 datasets, which we shall also use, and we achieve better accuracy using smaller networks and without pooling. We claim that our approach is also more efficient.

[HJL16] found that scaling scene images induced bias between training and testing sets, which significantly reduces performance. They proposed scale-specific networks in a multi-scale architecture. They also introduce a novel approach to combine pre-training on both the ImageNet and Places datasets, showing that more accurate classification is achieved. The authors mentioned the idea of redundant features by removing redundancy within the network, making it more efficient and forcing stronger regularization, thus improving generalization. Our proposed method develops this idea.

[LXM+18] combined two pre-trained CNNs in a hybrid collaborative representation method. One side of the hybrid model extracted shared features, while the other extracted class-specific features. They extended and improved their method by using various kernels: linear, polynomial, Hellinger and radial basis function. [ZTZ19] also used ImageNet networks pre-trained on VGG16 and Inception-v3 as feature extractors, before introducing a novel 3-layer additional network called CapsNet. The first layer is a convolutional layer that converts the input image into feature maps. The next layer consists of 2 reshape functions along with a squash function that transforms it into a 1-D vector before it enters the final layer, which has a node for each class, and is used for classification. Both these works use pre-trained networks, which our work shows can be significantly reduced in size.

[CLGY17] used VGG16 to extract important features, then used a method based on feature selection and feature fusion to merge relevant features into a final layer for classification. Following the work of [SZL+05] on canonical correlation analysis CCA, [CLGY17] improved their work by proposing discriminant correlation analysis, which over came the limitation of CCA that ignored the relationship between class structures in the data by maximizes the correlation between two feature sets while also maximizing the difference between the classes. [ZC19] adopted a Semantic Regional Graph model to select discriminant semantic regions in each image. The authors used a graph convolutional network originally proposed by [KW16], pre-trained on the COCO-Stuff dataset. This type of classification model showed great promise, especially on the difficult SUN397 data on which it achieved 74% classification accuracy.

[HLF18] introduced a novel way to tackle limited training data, which causes over-fitting in state-of-the-art CNNs such as AlexNet and VGG16. They used transfer learning, but also applied traditional augmentation on the orginal dataset, and collected images from the internet that were most similar to the desired

classes. This greatly increased the number of training examples and helped to prevent over-fitting. The authors showed that increasing the number of samples in the training data enables state-of-the-art networks to be trained on small datasets, such as scene recognition data. Our proposed method can be fine-tuned on the original data alone, with standard augmentation due to the strong regularization our pruning approach imposes on the networks.

[NPdS17] evaluated three common methods of using CNNs for remote sensing datasets with limited data: training a network from scratch, fine-tuning a pre-trained network, and using a pre-trained network as a feature extractor. The preferred method is training a network from scratch as it tends to generate better features and also allows for better control of the network. However, this approach is only feasible when adequate training data is available, otherwise either fine-tuning or feature extraction is more appropriate. The authors found that fine-tuning tends to lead to more accurate classification, especially when combined using a linear support vector machine as the classifier. Although our method uses a pre-trained network, we create more informative features by fine-tuning the full network, while applying our novel pruning approach to create a much smaller and more efficient network that helps to overcome over-fitting.

7.4 Materials and Methods

The robustness and performance of PulseNetOne was demonstrated using two state-of-the-art CNNs AlexNet and VGG16. It was evaluated on six benchmark datasets Aerial Images, MIT 67, NWPU-RESISC45, Scene15, SUN397 and UC Merced Land-Use datasets.

This section is broken down into three subsections: an outline of the experimental design, then a description of the classifiers and datasets used, and finally an explanation of our proposed algorithm.

7.4.1 Experimental Design

The proposed method was implemented in Python using the TensorFlow deep learning framework. The training and pruning phases were performed on a RTX2080 Ti NVIDIA graphics processing unit (GPU), while the inference stage was evaluated on both a RTX2080 GPU and an INTEL I7-8700K CPU with 32GB

RAM.

The six datasets had various size images which, for comparison with related work, were resized to 256×256 pixels. We used standard data-augmentation including random horizontal flipping, random adjustment of brightness and random increase/decrease of the image contrast. We also performed random cropping of the training dataset down to 224×224, and the test dataset was centrally cropped to 224×224. All images were then standardized by per-colour mean and standard deviation. The optimizer used for training the network is SGD, in conjunction with a step-wise decay learning rate schedule. The initial learning rate for both training and fine-tuning was 0.1, reduced by a factor of 10 when no decrease in loss on the validation set is detected for 20 epochs. The minimum learning rate used was 0.0001, and once there has been no improvement in the loss at that rate for 20 epochs the network is considered to have converged.

7.4.2 Convolutional Neural Networks

To evaluate the effectiveness of our proposed method, we run experiments using two state-of-the-art image recognition CNNs (AlexNet (see Section 5.5.4) and VGG16(see Section 5.5.5)) in various states, on all the datasets. The experiments are: train from scratch, transfer learning with models trained on Imagenet, fine-tune pre-trained Imagenet models, and finally pruning the fine-tuned models which are then further fine-tuned to regain accuracy as seen in Tables 7.2, 7.5, 7.6, 7.9, 7.10 and 7.12.

For transfer learning with models on Imagenet, the fully-connected layers are removed and new ones added. The convolutional layers are frozen and only the fully-connected layers are trained using the Adam optimizer with learning rate $1e^{-5}$. The fine-tuned pre-trained Imagenet models are the same as the transferred versions, except that all layers of the network are fine-tuned on the desired dataset. Finally, the pruned model is extracted from the fine-tuned model. Our proposed PulseNetOne method removes redundant filters/nodes from the fine-tuned model, leaving a compressed model.

7.4.3 Benchmark Remote Sensing and Scene Image Datasets

We evaluated our proposed PulseNetOne method on the benchmark datasets. These range in difficulty in terms of the scale of scene classes, images per

class, and diversity within the data. This means that on some datasets like UC Merced [YN10] 97%–99% classification accuracy is easily attainable using CNNs and could be considered the *MNIST* dataset of remote sensing research (an early dataset now considered trivial). Therefore, although we show our proposed method on these datasets, it is mainly to compare to other work. To demonstrate the potential of PulseNetOne we also use harder benchmark datasets such Remote Sensing Image Scene Classification (RESISC) [CHL17] and SUN397 [XHE+10]. The datasets are:

- The Aerial Image Dataset (AID) [XHH+17] contains 10,000 images with 30 classes including commercial, forest and church. The image resolution was 600×600 pixels but they were resized to 256×256 to compare with other works. The dataset was split with 50% randomly selected for the training set and 50% for the test set. A randomly chosen image from each class can be seen in Figure 7.1. It can be seen that some possible classes with similar attributes are meadow and forest, and industrial and commercial, which may cause some classifiers to misclassify samples in these categories.

- The MIT 67 dataset [QT09] has 67 classes representing types of indoor scene. It has 15,620 RGB images which were resized to 224×224 following the literature. The dataset was randomly divided into 80 samples (per class) for training and the remainder used for testing. To determine when the networks have converged, a validation dataset was needed. This was created using the training data with 60 images kept in the training set and 20 used for a validation set. Figure 7.2 shows a sample image from each of the 67 classes. Because of the large number of targets and some similar class types, this is one of the more difficult scene image classification tasks.

- The NWPU-RESISC45 dataset [CYY+18] consists of 31,500 remote sensing images with 45 categories. The images are of size 256×256 and there are 700 per class. As suggested in the literature, 20% of the samples within each class are used as the training dataset and the remaining 80% as the test set. Figure 7.3 shows an image from each class. Due to certain classes having much the same features (for example medium and dense residential, and railway and freeway) and because of the number of classes, this dataset has frequently been described as challenging for this research area.

- A small but popular dataset is Scene15 [LSP06], made up of natural and

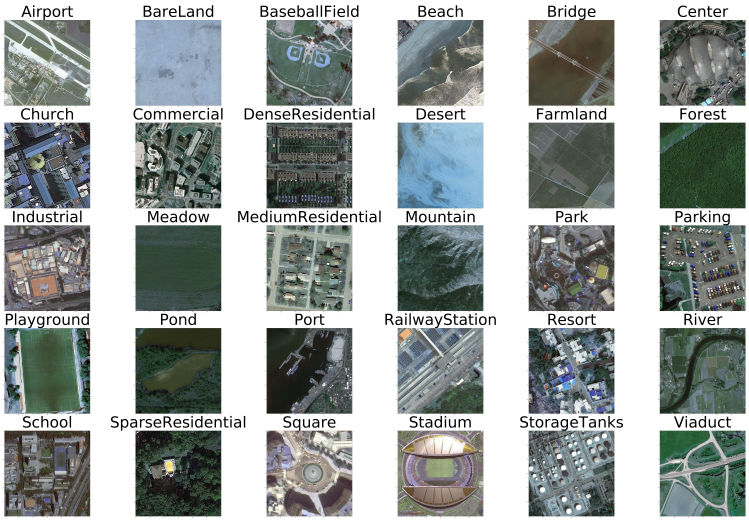

Figure 7.1: A random sample image of each class from the AID dataset.

Figure 7.2: A random sample image of each class from the MIT 67 dataset.

indoor categories with each class having between 210 and 410 examples. To compare with results in the literature, 100 random samples from each class are used for training and the rest for testing. The training dataset is split further into 80 images per class for training and 20 for validation. A sample image from each class is shown in Figure 7.4. Although this is the oldest dataset used in this work, it still is a good test for the robustness of

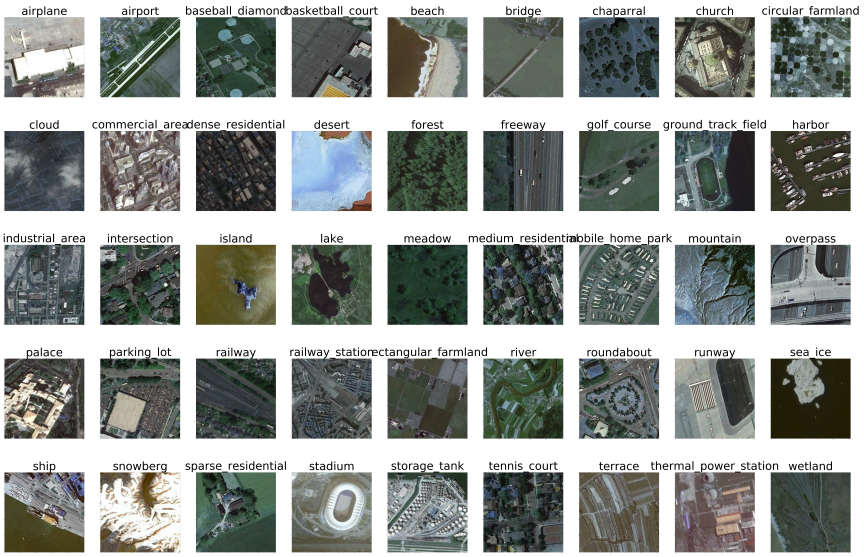

Figure 7.3: A random sample image of each class from the NWPU-RESISC45 dataset.

our method, and has a great deal of related work for comparison.

Figure 7.4: A random sample image of each class from the Scene15 dataset.

- The SUN397 dataset [XHE+10] is a much larger dataset made up of 108,754 images with 397 classes, ranging from 100 to 2,361 examples per class, and including natural, indoor and man-made images. Only 40 samples are used for training of each class, 10 images per class for the validation set and the rest for the test set. Due to the large number of classes within the dataset, Figure 7.6 only shows a sample from the first 100 classes. For this work, and as reported by others, this was the most challenging remote sensing dataset for two main reasons: the large number of classes and the small training set.

- The UC Merced Land-Use dataset (UCM) [YN08] has 2,100 scene images taken from above, separated into 21 classes with 100 examples of 256×256 pixels in each class. The dataset was constructed from aerial orthoimagery in various US regions including Dallas, Tampa and Las Vegas, with categories including runway, golf course and different degrees of residential density. Figure 7.5 displays a random image from each of the 21 classes, and although it is the smallest dataset, with similar classes such as dense and medium residential, and freeway and runway, this dataset was relatively simple for a CNN. This is perhaps because it has the highest training-to-test set ratio, which gives the network more images to be trained on, and less for the evaluation part.

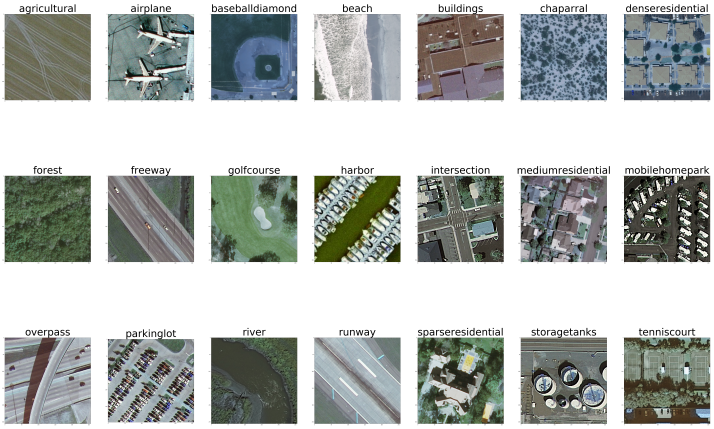

Figure 7.5: A random sample image of each class from the UC Merced Land-Use dataset.

Due to the limited number of training samples, to compute the final model accuracy the validation set is merged with the training set to recreate the original training data, and the model is further trained for a few epochs [TWK17]. The experiments are validated using 5-fold cross validation, randomly selecting the training and testing data splits at each fold. All images were resized to 256×256 where necessary. Also due to the limited number of training samples, we use transfer learning, a common and very effective technique used in deep learning. We take both networks, AlexNet and VGG16, already trained on the big-data dataset ImageNet that has over a million training images with 1000 different classes. By using these trained networks as a *good* starting for the weight initialization, and then fine-tune the networks with the targeted dataset, we achieve much better classification accuracy, similar to other work in this area. We only retain the trained weights in the convolutional layers, as the fully-connected layers are more class specific to the dataset it was trained on.

Dataset	# of Classes	Ímages per class	Total Images	Train/Test split	Image Sizes	Year
Scene-15 [LSP06]	15	210-410	4485	100 imgs/rest	Multi Scales	2006
MIT-67 [QT09]	67	101-734	15620	80 imgs/rest	Multi Scales	2009
UC Merced [YN08]	21	100	2100	80%/20%	256×256	2010
SUN397 [XHE+10]	397	100-2361	10000	40 imgs/rest	Multi Scales	2010
NWPU-RESISC45 [CHL17]	45	700	31500	20%/80%	256×256	2016
AID [XHH+17]	30	220-420	10000	20%/80%	600×600	2016

Table 7.1: Comparison of the six remote sensing datasets used in this work.

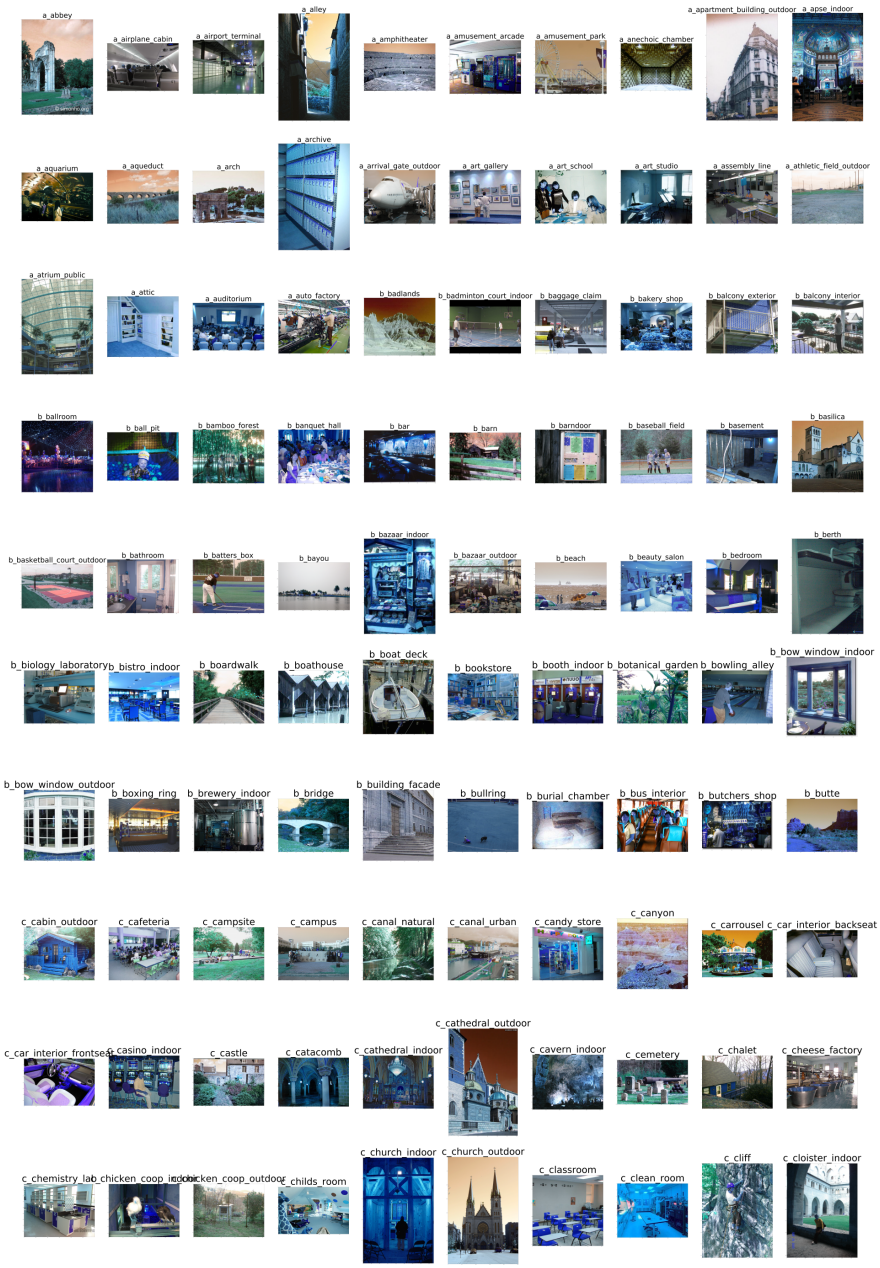

Figure 7.6: A random sample image of each class from the SUN397 dataset.

7.4.4 PulseNetOne

Our proposed method PulseNetOne (Algorithm 9) is quite similar to unsupervised PulseNet (see Section 6.4.3), with 2 main differences. Firstly, where unsupervised PulseNet (Algorithm 7) has a limit on how much of each layer can be pruned (25%), for PulseNetOne we turn off this limit and allow it to cluster and prune as much or as little it sees fit to. The second difference is that instead of using the coarse, medium and fine pruning as in unsupervised PulseNet, we only use coarse pruning and for a single iteration. This makes the training pro-

cess extremely fast, resulting in a rapidly pruned network. Algorithm 9 requires less initial parameters than Algorithm 7, and iteratively moving through all the layers (parameter *L* in Algorithm 9) in the network, prunes each layer individually. The method used to determine the important filters is Algorithm 8, the same as that used in unsupervised PulseNet.

The following bullet points give a line-by-line description of Algorithm 9:

- In line 1 we initialize p_{in} to represent the 3 channels of an RBG image.
- Lines $2-20$ are a loop where we loop through all the layers of the network.
- In line $3-10$ is an if condition that checks if we are analysing a convolution layer.
- Line 4 defines the filter matrix of the layer as a 4-D matrix of shape $w,\ x,\ y,\ z$.
- In line 5, w and x is the shape of the kernal in the layer (3×3), and y represents the input shape and z is the output shape. We remove the rows in y that are not in p_{in}.
- Line 6 we reshape the 4-D matrix into a 2-matrix without changing its data.
- Line 7 using Algorithm 8 to return the index of important rows, which are the clusters centers, as p_{out}.
- In line 8 we remove the rows in z that are not in p_{out}.
- Line 9 we reshape the 2-D matrix back into a 4-matrix without changing its data.
- In line $11 - 18$ is an if condition that checks if we are analysing a fully-connected layer.
- Line 12 defines the node matrix of the layer as a 2-D matrix of shape $m,\ n$.
- In line 13 we remove the rows in m that are not in p_{in}.
- Line 14 we transpose the 2-D matrix by rotating it around its axis, without changing its data.
- Line 15 using Algorithm 8 to return the index of important rows, which are the clusters centers, as p_{out}.
- In line 16 we remove the rows in n that are not in p_{out}.

- Line 17 we transpose the 2-D matrix back into its original shape by rotating it around its axis, without changing its data.
- In line 19 the desired output cluster centers of the previous layer becomes the input clusters of the following layer. Therefore we set p_{in} to p_{out}.
- Line 21 we fine-tune the network to convergence.

Algorithm 9 PulseNetOne K-means Algorithm

```
1: initialize set p_in = {0, 1, 2}
2: for all l ∈ L do
3:     if l == convolution layer then
4:         given filter F represented by (w, x, y, z)
5:         Remove rows in y not in p_in
6:         Transpose (w, x, y, z) → (z, w * x * y)
7:         p_out =Get optimal k clusters of (z, w * x * y) (Algorithm. 8)
8:         Remove rows in z not in p_out
9:         Transpose (z, w * x * y) → (w, x, y, z)
10:    end if
11:    if l == fully connected layer then
12:        given node N represented by (m, n)
13:        Remove rows in m not in p_in
14:        Transpose (m, n) → (n, m)
15:        p_out =Get optimal k clusters of (m, n) (Algorithm. 8)
16:        Remove rows in n not in p_out
17:        Transpose (n, m) → (m, n)
18:    end if
19:    p_in ← p_out
20: end for
21: Fine-Tune Network until it converges
```

7.5 Results

PulseNetOne was applied to the AID dataset and its performance compared with other state-of-the-art results, as shown in Table 7.3. Table 7.2 shows that training the networks on the target dataset from scratch yielded poor results: 61.24% and 56.67% on AlexNet and VGG16 respectively. The accuracy of both networks improved greatly when using transfer learning and fine-tuning the transferred

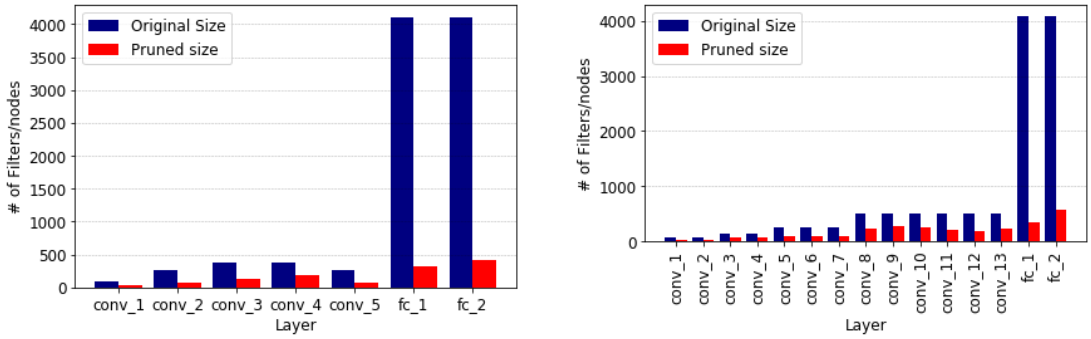

Figure 7.7: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the AID dataset.

model. For transfer learning the weights and parameters of networks trained on ImageNet, on both AlexNet and VGG16, were re-initialized and used as a starting point for the weights of our networks. The fully-connected layers of the pre-trained networks were removed and replaced by fully-connected layers of the same size initialized using a *He normal* weight distribution (where the samples are drawn from a truncated normal distribution centred at 0 and a standard deviation related to the number of filters in trhe layer) ([HZRS15]), with the number of final layer outputs being the number of classes in the dataset.

PulseNetOne takes the fine-tuned network and, as described in Section 7.4.4, prunes the original network to a much smaller version, which not only reduces the storage size and number of floating point operations per second (FLOPs), but also improves classification accuracy. The accuracy of AlexNet improves by nearly 5.5% and VGG16 by 3.5%. The confusion matrices of both networks in Figures 7.9 and 7.10 show that very few errors were made, and as they were quite randomly distributed they might simply be a poor representation of the class within wrongly-classified images. The precision, recall and F1 scores of both networks are in their descriptions for further comparison.

Figure 7.7 shows the number of filters pruned in each layer of the networks. As expected, the layers pruned most are the fully-connected layers, as it is well documented that these are over-parameterized. It is interesting to see that the first and last few convolutional layers are pruned more than the intermediate layers. A reason for this is that the first few layers are edge detectors and filters to based on colour and shapes which can contain a lot of duplicate or similar filters, while the last convolutional layers are more class-related and, because the networks were pre-trained on the 1000-class ImageNet dataset, these layers contained many redundant filters.

Table 7.14 shows inference time and energy consumed per image on both a CPU and GPU processor, along with the storage size of the original and compressed networks. To ensure that the work was application related, we used a test batch of just one image which, is more applicable to real-world testing. The storage sizes of both networks are greatly reduced by PulseNetOne, and on a CPU inference is approximately $3\times$ faster, and approximately $10\times$ faster on a GPU. The energy saving for the networks ranges from $1.5\times$ to $13\times$ fewer milli-Joules. The bar-charts in Figure 7.8 clearly show that, although in most cases the theoretical improvements are not reached (except for the GPU timings for both networks, and the GPU energy for AlexNet), the results come close in most cases.

Comparing our work with state-of-the-art results in Table 7.3, it can be seen that our approach using VGG16 achieves the best classification accuracy with 99.77%, and our AlexNet version ranks in second place with 98.91%, which outperforms both Discriminative CNNs VGG16 [CYY+18] and GCFs+LOFs [CZT+18] by nearly 3%.

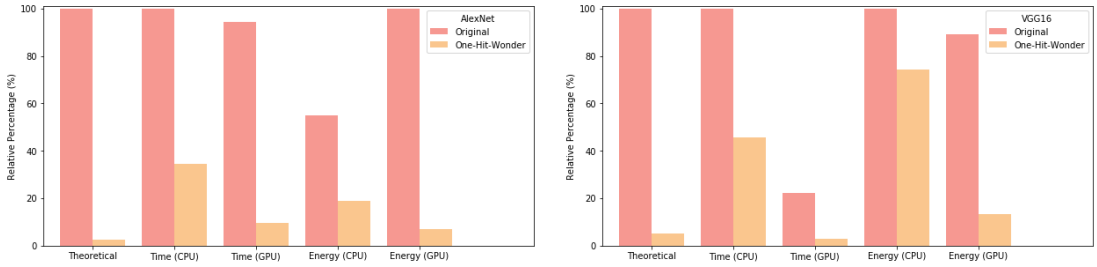

Figure 7.8: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the AID dataset.

Next the dataset MIT67 is analysed, on which (similarly to the AID dataset) CNNs achieve poor classification accuracy when trained from scratch: 32.31% accuracy on AlexNet and 26.57% on VGG16. The explanation is believed to be the lack of training samples: a deep learning network has millions of parameters to tune and is therefore quite data-hungry. Table 7.5 shows that simply transferring learning was not as effective as on the previous dataset, reaching a maximum accuracy of nearly 70%, but after fine-tuning on the targeted dataset, it reached a more reasonable 92.74%. PulseNetOne reduces AlexNet down to 2.28% and VGG16 down to 8.26% of their original sizes, and improves their performances to 95.83% and 96.68% respectively.

The confusion matrices for both networks, in Figures 7.12 and 7.13, show that

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
Scratch	$96-256-384-384-256-4096-4096$	58404254 (100%)	116789320(100%)	61.24±1.53
Transferred	-	-	-	91.17±0.10
Fine tuned	-	-	-	93.51±0.12
PulseNetOne	$36-66-135-180-79-317-409$	1544061 (2.64%)	3085626(2.64%)	**98.91±0.07**
VGG16				
Scratch	$64-64-128-128-256-256-256-512$ $-512-512-512-512-4096-4096$	134383454 (100%)	268742032(100%)	56.67±0.31
Transferred	-	-	-	93.64±0.26
Fine tuned	-	-	-	96.11±0.19
PulseNetOne	$23-28-60-59-90-95-103-228$ $-267-243-203-174-230-354-570$	6942627 (5.17%)	13879756(5.16%)	**99.77±0.06**

Table 7.2: Overall accuracies and standard deviations (%) of different types of CNNs methods along with PulseNetOne on the AID dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

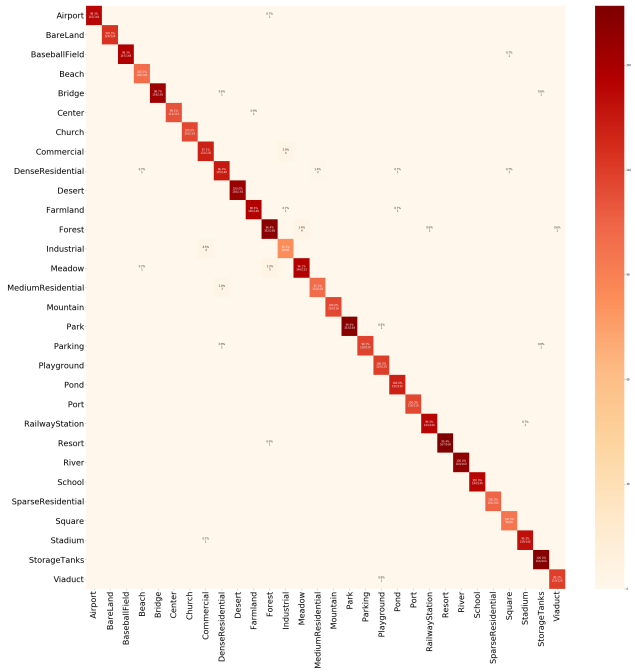

Figure 7.9: The confusion matrix of the first fold from the 5-fold cross-validaiton on the AID dataset using the PulseNetOne pruned AlexNet model. It has an accuracy score of 98.95%, a precision score of 98.96%, recall score of 98.95% and a F1-score of 98.95%.

most mistakes were made when distinguishing between the bathroom and bedroom, and the grocery store and toy store classes. The precision, recall and F1 scores of both networks are in their descriptions for further transparency, and can be seen to be between 95.89% and 96.92%. Details of how PulseNetOne pruned the layers of the networks are shown in Figure 7.14, and agree with the analysis of the AID dataset. However, the VGG16 network retained more nodes in the fully-connected layers, which could be caused by the MIT67 dataset having more than twice the number of target classes.

Table 7.14 shows computational efficiency results for both the original and compressed networks. The storage size of AlexNet was reduced in size by nearly $44\times$ while VGG16 was reduced by $12X$. The energy saved on the CPU and GPU was $11\times$ and 5.5–$11\times$ respectively, while the speed-up in inference time on AlexNet was 3–$10\times$ and on VGG16 3–$7\times$.

Table 7.4 compares PulseNetOne to the related work in this area, and it can be seen that both networks pruned by PulseNetOne outperform the state-of-the-art by over 6%. FOSNet CCG [SHK19] and SOSF+CFA+GAF [SLL+18] were the current best published results on the MIT67 dataset, achieving 90.37% and 89.51% respectively, but were significantly beaten by PulseNetOne. Figure 7.11

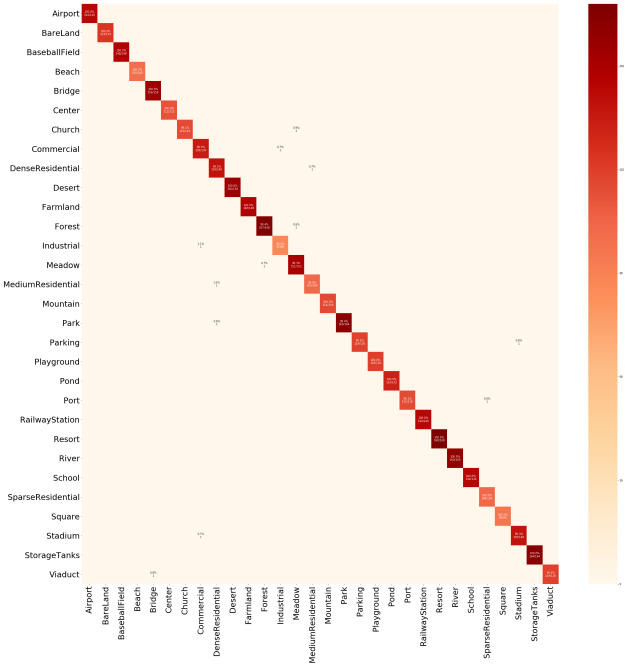

Figure 7.10: The confusion matrix of the first fold from the 5-fold cross-validation on the AID dataset using the PulseNetOne pruned VGG16 model. It has an accuracy score of 99.70%, a precision score of 99.70%, recall score of 99.70% and a F1-score of 99.70%.

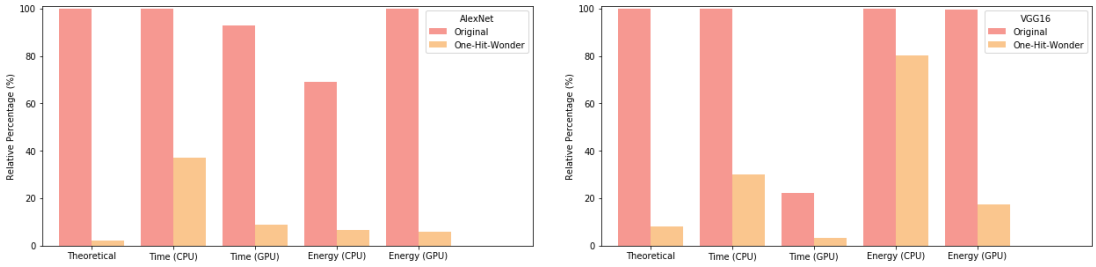

Figure 7.11: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the MIT67 dataset.

shows that AlexNet almost achieved its theoretical performance on all experiments except for CPU inference timing, while the pruned network was approximately $3\times$ faster than the original network. VGG16 results were more mixed, with the CPU timing beating the theoretical result, but the CPU energy usage being quite high, though still slightly less than the original network structure.

The NWPU-RESISC45 dataset accuracy was only able to score 27.11% on AlexNet and 17.87% on VGG16 when trained from scratch. Table 7.6 shows that transfer learning boosted their performances up to approximately 79% accuracy, while fine-tuning increased both to approximately 84%. PulseNetOne was able to in-

Table 7.3: A comparison between state-of-the-art and PulseNetOne results on the AID data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
CaffeNet [XHH+17]	2017	89.53
VGG-VD-16 [XHH+17]	2017	89.64
Fusion by addition [CLGY17]	2017	91.87
Discrimative CNNs AlexNet [CYY+18]	2018	94.47
Two-Stream Fusion [YL18]	2018	94.58
VGG16-CapsNet [ZTZ19]	2019	94.74
Discrimative CNNs GoogLeNet [CYY+18]	2018	96.22
Inception-v3-CapsNet [ZTZ19]	2019	96.32
GCFs+LOFs [CZT+18]	2018	96.85
Discrimative CNNs VGG16 [CYY+18]	2018	96.89
AlexNet-PulseNetOne	2020	**98.91**
VGG16-PulseNetOne	2020	**99.77**

crease their classification accuracy by over 10%, with AlexNet scoring 94.65% and VGG16 94.86%. This was the result of network pruning reducing overfitting: AlexNet was reduced by $67\times$ and VGG16 by $33\times$.

The confusion matrices for both networks, in Figures 7.17 and 7.18, show that both networks found it hard to distinguish between the freeway and railway, medium and dense residential, and meadow and forest classes. Other publications have commented on these classes being difficult to separate, and the examples in Figure 7.3 illustrate that they have similar features. Again, the precision, recall and F1 scores of both networks are given in their descriptions, and can be seen to be between 94.65% and 94.89%. The way in which PulseNetOne pruned the layers of the networks, shown in Figure 7.16, is more similar to that of the AID dataset than the MIT67 dataset, possibly because of its 45 classes. The storage sizes of AlexNet and VGG16 were reduced by $44\times$ and $33\times$ respectively, as shown in Table 7.14. The speed-up in inference time on both networks were between $3\times$ and $10\times$ on the CPU and GPU, respectively.

Figure 7.15 shows that once again AlexNet achieved close to its theoretical performance on all experiments except for the CPU inference timing. The VGG16 results were not quite as impressive, with the GPU timing beating the theoretical result, while the CPU energy usage was close to that of the original network. It can be seen from Figure 7.15 that in the other experiments the pruned network

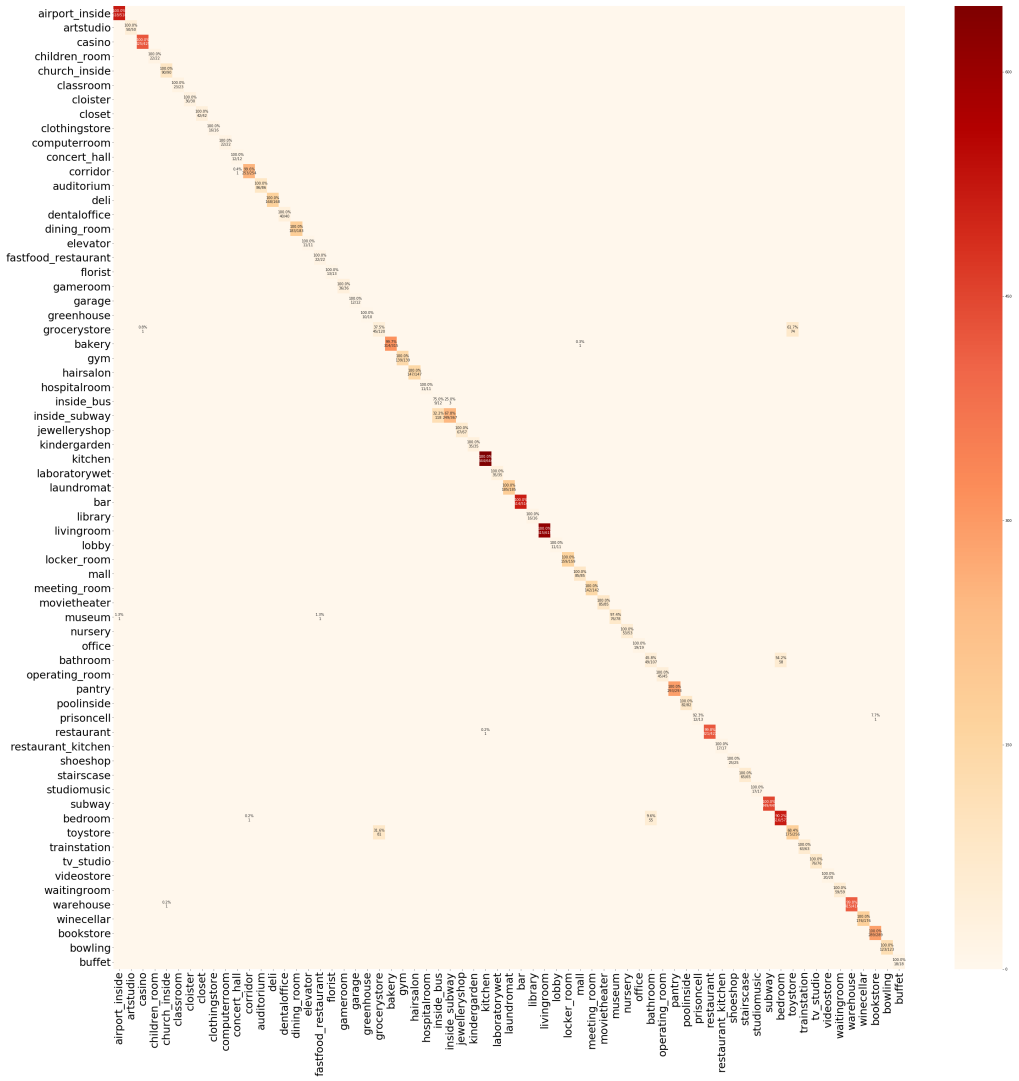

Figure 7.12: The confusion matrix of the first fold from the 5-fold cross-validation on the MIT67 dataset using the PulseNetOne pruned AlexNet model. It has an accuracy score of 95.89%, a precision score of 96.07%, recall score of 95.89% and a F1-score of 95.90%.

is much more efficient than the original.

PulseNetOne again beats the current state-of-the-art on the NWPU-RESISC45 dataset, as seen in Table 7.7, by just over 2%. Inception-v3-CapsNet [ZTZ19] and Triple networks [LH17] achieve 92.60% and 92.33% respectively, which is quite close to our results, but it should be noted that PulseNetOne creates an extremely efficient version of the networks, while both the related works approaches use complex networks that increase computational expense.

The accuracy achieved on Scene15 dataset, when trained from scratch, was quite reasonable when compared to the previous datasets, with AlexNet scor-

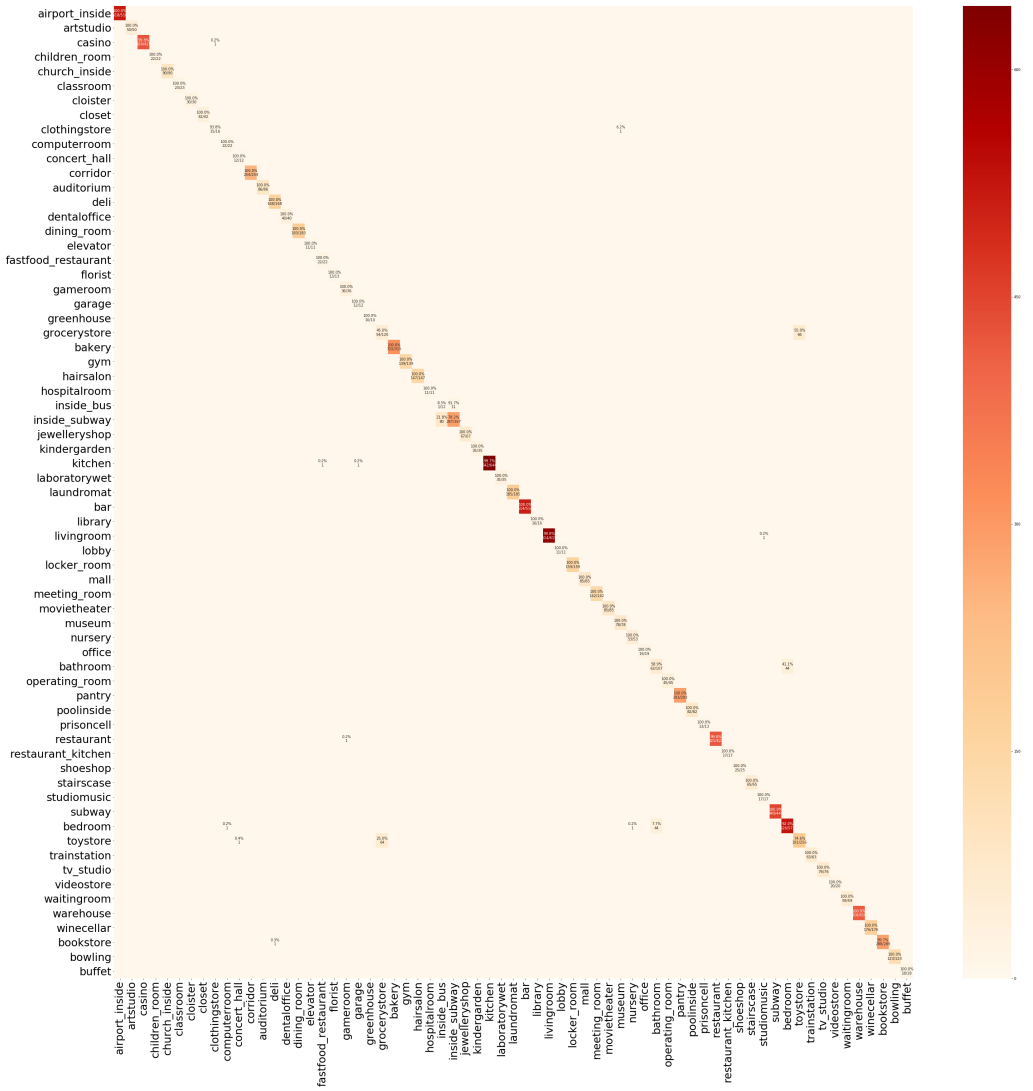

Figure 7.13: The confusion matrix of the first fold from the 5-fold cross-validation on the MIT67 dataset using the PulseNetOne pruned VGG16 model. It has an accuracy score of 96.69%, a precision score of 96.92%, recall score of 96.69% and a F1-score of 96.72%.

ing 77.91% and VGG16 scoring 79.69%. Transfer learning increased classification accuracy by 10–12%, while fine-tuning further increased it by 2–4% as seen in Table 7.6. PulseNetOne was able to increase the classification accuracy of AlexNet to 97.96% and VGG16 to 97.48%. AlexNet was reduced to 1.3% and VGG16 to 3.04% of their original sizes. The confusion matrices of both networks Figures 7.19 and 7.21 show no particular pattern in their errors, with both networks making random mis-classifications. The precision, recall and F1 scores of both networks are given in their descriptions, and can be seen to be between 97.46% and 98%.

Table 7.4: A comparison between state-of-the-art and PulseNetOne results on the MIT67 data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
Otc and HOG [MZMT14]	2014	47.33
AlexNet fine-tuned on Imagenet [ZLX+14]	2014	56.79
AlexNet fine-tuned on Place205 [ZLX+14]	2014	68.24
Hybrid-CNN [ZLX+14]	2014	70.80
GoogLeNet fine-tuned on Imagenet [SLJ+15]	2014	72.31
DSFL CNN [ZWS+14]	2017	76.23
GoogLeNet fine-tuned on Place205 [SLJ+15]	2014	77.54
DSP [GWWL15]	2014	78.28
InterActive [XZW+16]	2016	78.65
G-MS2F (ADD) [TWK17]	2017	79.18
G-MS2F (Prod) [TWK17]	2017	79.63
SDO [CYY+18]	2018	86.76
MP [SJH17]	2017	86.90
Sparse Representation [NLB+17]	2017	87.22
MFAFVNet+Places [LDV17]	2017	87.97
SRG [ZC19]	2019	88.13
SOSF+CFA+GAF [SLL+18]	2018	89.51
FOSNet CCM-CCG [SHK19]	2019	90.30
FOSNet CCG [SHK19]	2019	90.37
AlexNet-PulseNetOne	2020	**95.83**
VGG16-PulseNetOne	2020	**96.68**

The layers of both networks, as shown in Figure 7.20, are pruned in the same pattern as with the other datasets, fully-connected layers being heavily pruned, along with the beginning and ending of the convolutional layers, while the intermediate convolutional layers are less pruned. The storage sizes of AlexNet and VGG16 were reduced by $77\times$ and $33\times$ respectively, as seen in Table 7.14. The improvement of the inference timing on both networks were $3\times$ and $12\times$ for the CPU and GPU, respectively. Table 7.14 also shows that the CPU energy saving is between $2\times$ and $4\times$, and between $14.5\times$ and $16\times$ for the GPU.

Figure 7.22 shows that on AlexNet the compressed network performs better on the GPU, but for real-world situations where a CPU would be more commonly used for analysis, the pruned network easily outperforms the original structure in all cases. On VGG16 the CPU energy usage are unimpressive, but are almost twice as energy efficient as the original network. Following the same analysis as for the previous datasets, we compare PulseNetOne to the current state-of-the-

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	$96-256-384-384-256-4096-4096$	58555843 (100%)	117092424(100%)	32.31±2.36
Transferred	-	-	-	63.45±0.61
Fine-tuned	-	-	-	91.84±1.28
PulseNetOne	$35-118-132-133-67-317-204$	1338355 (2.29%)	2674572(2.28%)	**95.83±0.11**
VGG16				
scratch	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	134535043 (100%)	269045136(100%)	26.57±1.38
Transferred	-	-	-	69.92±2.08
Fine-tuned	-	-	-	92.74±1.31
PulseNetOne	$28-15-32-36-104-100-91-143-154-163-173-144-220-826-752$	11111636 (8.26%)	22217192(8.26%)	**96.68±0.62**

Table 7.5: Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the MIT67 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	$96-256-384-384-256-4096-4096$	58465709 (100%)	116912200(100%)	27.11±2.63
Transferred	-	-	-	79.86±0.18
Fine-tuned	-	-	-	83.29±0.25
PulseNetOne	$30-76-115-62-58-257-231$	850336 (1.45%)	1698932(1.45%)	**94.65±0.95**
VGG16				
scratch	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	134444909 (100%)	268864912(100%)	17.87±3.47
Transferred	-	-	-	78.85±0.16
Fine-tuned	-	-	-	84.36±0.22
PulseNetOne	$21-20-53-45-88-109-74-230-165-194-200-199-137-300-268$	4074009 (3.03%)	8143738(3.03%)	**94.86±1.03**

Table 7.6: Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the NWPU-RESISC45 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

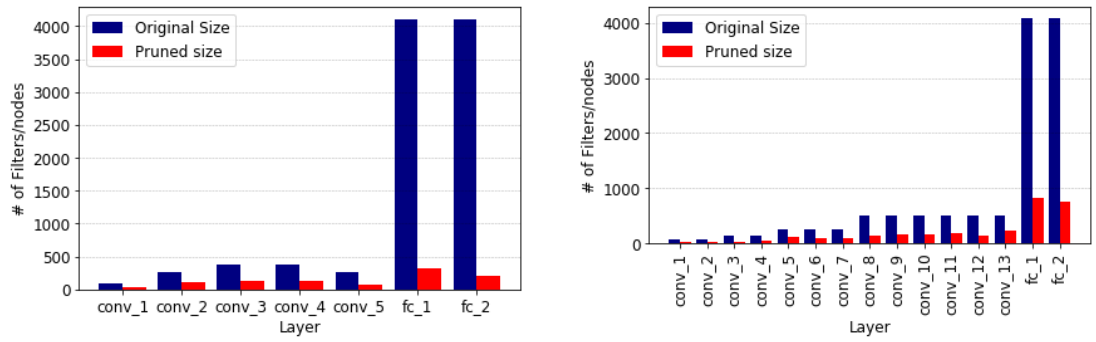

Figure 7.14: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the MIT67 dataset.

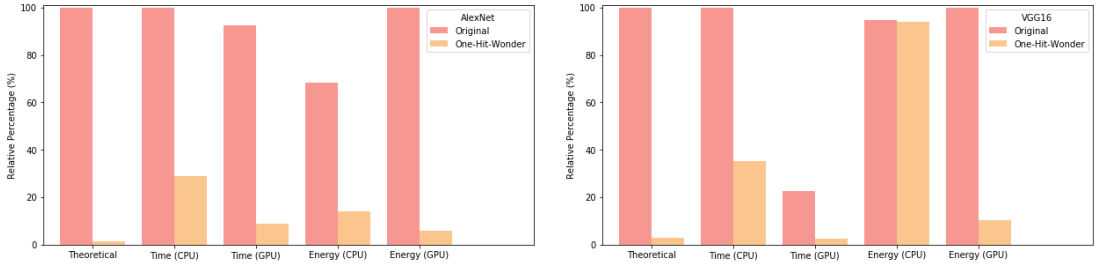

Figure 7.15: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the NWPU-RESISC45 dataset.

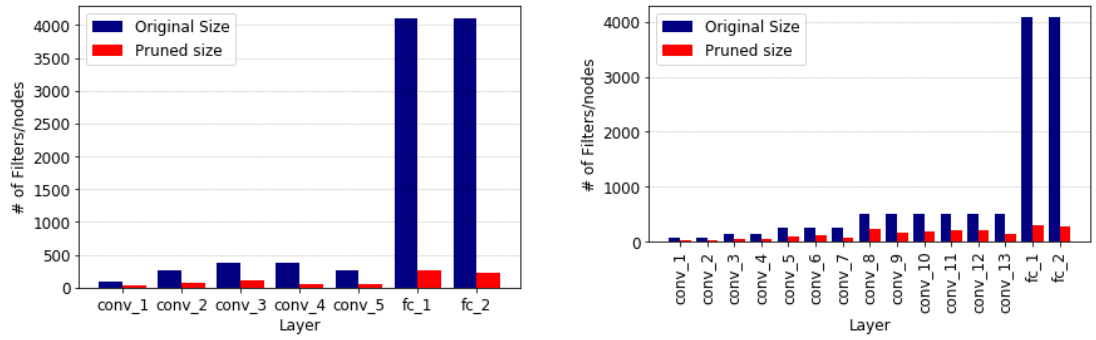

Figure 7.16: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the NWPU-RESISC45 dataset.

art on the Scene15 dataset, shown in Table 7.8. The Dual CNN [HJL16] was state-of-the-art since 2016, with a classification accuracy of 95.18%, the next in line being G-MS2F [TWK17] with 92.90%. PulseNetOne achieves almost a 3% greater accuracy, with AlexNet beating VGG16 on this dataset with 97.97%. Again, PulseNetOne's closest competition uses 2 deep learning CNNs which is much less efficient.

The SUN397 dataset was the most difficult dataset to train from scratch, possible because of its large number of classes and its limited amount of training

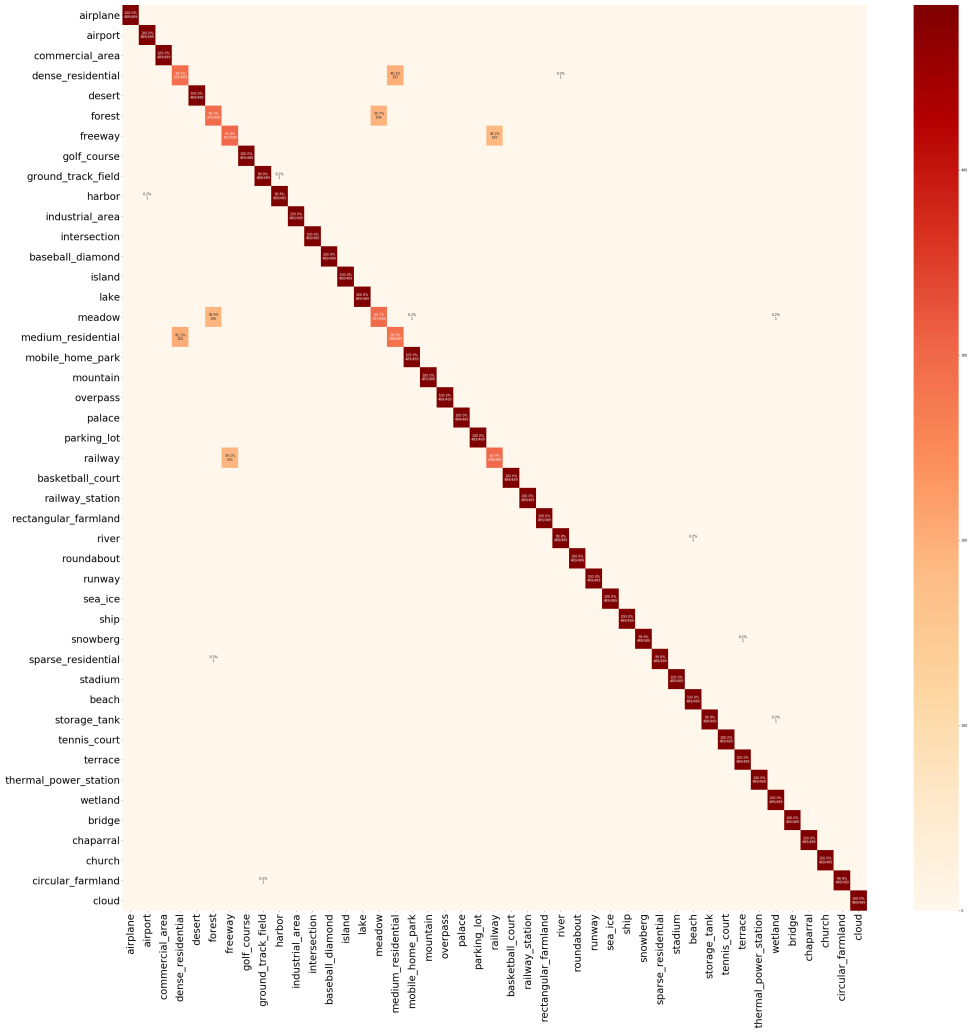

Figure 7.17: The confusion matrix of the first fold from the 5-fold cross-validation on the NWPU-RESISC45 dataset using the PulseNetOne pruned AlexNet model. It has an accuracy score of 94.65%, a precision score of 94.68%, recall score of 94.65% and a F1-score of 94.66%.

data: AlexNet reached 21.24% and VGG16 15.41% accuracy. Transfer learning with the ImageNet dataset helped increased AlexNet's accuracy to 42.91% and VGG16 to 41.51%. Fine-tuning all the layers for either network had less effect than with the other datasets, only improving AlexNet to 49.89% and VGG16 to 50.29%. However, PulseNetOne was able to pass 80% classification accuracy on both networks, as seen in Table 7.10: AlexNet reached 82.11% accuracy and VGG16 84.32%. Table 7.11 shows previous state-of-the-art results, and our method advances the best by approximately 5%. SOSF+CFA+GAF [SLL+18] achieves the closest to our results, but whereas we use an input image of size 256×256, their method uses an input size of 608×608 which is significantly larger and therefore more computationally expensive. FOSNet [SHK19] is once

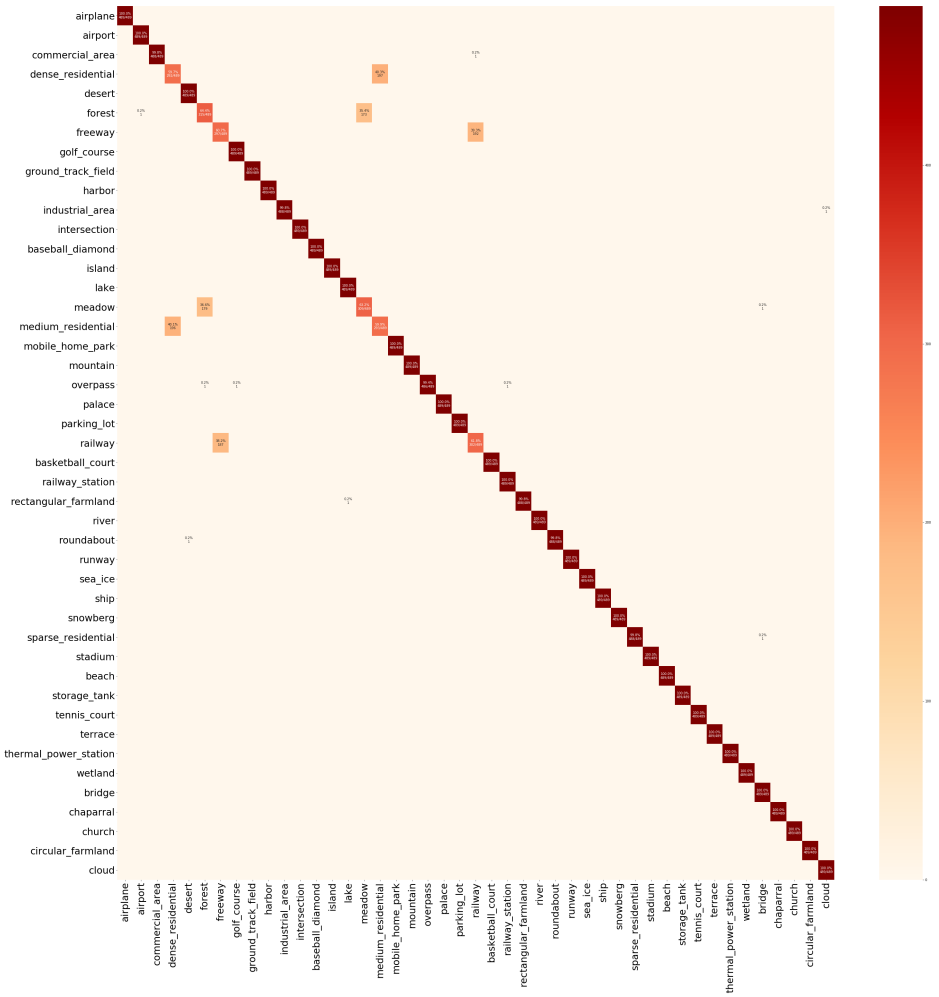

Figure 7.18: The confusion matrix of the first fold from the 5-fold cross-validation on the NWPU-RESISC45 dataset using the PulseNetOne pruned VGG16 model. It has an accuracy score of 94.87%, a precision score of 94.89%, recall score of 94.87% and a F1-score of 94.88%.

again close to the state-of-the-art, with 77.28%.

Confusion matrices for both networks were created, but are not shown for space reasons. There were no real surprises in either of them, similar classes being mis-classified. The PulseNetOne-pruned AlexNet model had a precision score of 85.60%, recall score 82.45% and F1-score 83.24%, and the VGG166 version had precision score 86.80%, recall score 84.29% and a F1-score 84.93%. The layers of both networks, as seen in Figure 7.23, are similar to previous results, but show that AlexNet's last convolution layer was heavily pruned: slightly surprising given the dataset's large number (397) of classes. A possible explanation is that the classes in the ImageNet dataset, on which the model was initially trained, were quite different to the classes on the target dataset SUN397. The

Table 7.7: A comparison between state-of-the-art and PulseNetOne results on the NWPU-RESISC45 dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
Two-Stream Fusion [YL18]	2018	83.16
Fine-tuned CNNs AlexNet [CHL17]	2017	85.16
Fine-tuned CNNs GoogLeNet [CHL17]	2017	86.02
Discrimative CNNs AlexNet [CYY$^+$18]	2018	87.24
VGG16-CapsNet [ZTZ19]	2019	89.18
Fine-tuned CNNs VGG16 [CHL17]	2017	90.36
Discrimative CNNs GoogLeNet [CYY$^+$18]	2018	90.49
Discrimative CNNs VGG16 [CYY$^+$18]	2018	91.89
Triple networks [LH17]	2017	92.33
Inception-v3-CapsNet [ZTZ19]	2019	92.60
AlexNet-PulseNetOne	2020	**94.65**
VGG16-PulseNetOne	2020	**94.86**

PulseNetOne-pruned VGG16 also had a convolutional layer that was heavily pruned (the $12th$ layer or second-to-last layer), which reaffirms our hypothesis.

The storage sizes of AlexNet and VGG16 were reduced $54\times$ and $87\times$ respectively, as shown in Table 7.14. The improvement in inference timing on both networks were $3.5\times$ and $10\times$ for the CPU and GPU, respectively. As shown in Table 7.14 the CPU energy saving is 2–$3\times$ while the GPU's energy saving is 13.5–$17\times$. Figure 7.24 shows that the PulseNetOne version of both networks use considerably less energy and are in all cases noticeably faster at inference time.

The final dataset is UC Merced, which (as mentioned earlier) is relatively easy to classify accurately. VGG16 and AlexNet achieve 75.01% and 83.25% respectively, while transfer learning applied only to the new fully-connected layers resulted in accuracies of approximately 95%. Further fine-tuning increased the accuracies of both networks to just over 98%, and the current state-of-the-art uses fine-tuning with a Support Vector Machine as the classifier. PulseNetOne added 1.5% accuracy to both networks, and though this is small it makes our proposed method state-of-the-art. Table 7.12 shows that VGG16 achieved classification accuracy 99.69%, and AlexNet 99.82%. We attribute this improvement to the high degree of pruning reducing over-fitting: the AlexNet parameters and FLOPS were reduced by $30\times$, and those of VGG16 by $71.5\times$.

Table 7.8: A comparison between state-of-the-art and PulseNetOne results on the SCENE15 data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy (%)
AlexNet fine-tuned on Imagenet [ZLX+14]	2014	84.23
Otc and HOG [MZMT14]	2014	84.37
LGF [ZLCD16]	2016	85.80
GoogLeNet fine-tuned on Imagenet [SLJ+15]	2014	91.12
AlexNet fine-tuned on Place205 [ZLX+14]	2014	90.19
Hybrid-CNN [ZLX+14]	2014	91.59
DSP [GWWL15]	2014	92.16
GoogLeNet fine-tuned on Place205 [SLJ+15]	2014	92.16
G-MS2F (ADD) [TWK17]	2017	92.70
DSFL CNN [ZWS+14]	2017	92.81
G-MS2F (Prod) [TWK17]	2017	92.90
Dual CNN [HJL16]	2016	95.18
VGG16-PulseNetOne	2020	**97.48**
AlexNet-PulseNetOne	2020	**97.96**

The confusion matrices of both networks, seen in Figures 7.26 and 7.27, show that between both networks there was only 3 mis-classifications. The precision, recall and F1 scores of both networks are in their descriptions, and can be seen to be 99.36–99.70%. As expected, the GPU performance was close to theoretical, while CPU times were significantly better on the pruned networks compared to the original networks, as shown in Table 7.14.

Figure 7.28 shows that the second-last layer in VGG16 is again the most compressed, as in the SUN397 dataset, with the other layers being pruned in the same pattern as the other datasets. The storage sizes of AlexNet and VGG16 were reduced from 222.654MB to 7.404MB and 512.492MB to 7.195MB, respectively. The speed-up in inference time on both networks were $3.5\times$ and $12\times$ on the CPU and GPU, respectively. Figure 7.25 shows that the GPU was more energy efficient and also had a faster inference time, but the PulseNetOne versions of the networks were both faster and consumed less energy on the CPU than the original networks. PulseNetOne slightly outperforms the current state-of-the-art on the UC Merced dataset by almost 0.5%, as shown in Table 7.13. Inception v3 CapsNet [ZTZ19] had, once again, one of the best accuracies with 99.05%, narrowly beaten by Fine-tuned GoogLeNet with SVM [NPdS17] with 99.47% which, though quite close to our results, comes at the cost of more com-

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	$96-256-384-384-256-4096-4096$	58342799 (100%)	116666440(100%)	77.91±0.85
Transferred	-	-	-	87.52±0.38
Fine-tuned	-	-	-	91.68±0.72
PulseNetOne	$36-90-168-53-32-301-274$	759853 (1.30%)	1517776(1.30%)	**97.96±0.62**
VGG16				
scratch	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	134321999 (100%)	268619152(100%)	79.69±1.27
Transferred	-	-	-	89.75±1.01
Fine-tuned	-	-	-	92.84±0.48
PulseNetOne	$10-22-36-36-65-68-76-109-168-198-167-167-169-305-257$	4079549 (3.04%)	8155378(3.04%)	**97.48±0.81**

Table 7.9: Overall accuracies and standard deviations (%) of different CNNs methods along with the proposed PulseNetOne on the Scene15 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	$96-256-384-384-256-4096-4096$	59907853 (100%)	119795784(100%)	21.24±1.98
Transferred	-	-	-	42.91±1.71
Fine-tuned	-	-	-	49.89±1.18
PulseNetOne	$35-103-176-160-41-242-264$	1105768 (1.85%)	2208708(1.84%)	**82.11±2.48**
VGG16				
scratch	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	135887053 (100%)	271748496(100%)	15.41±1.75
Transferred	-	-	-	41.51±3.84
Fine-tuned	-	-	-	50.29±2.89
PulseNetOne	$16-27-35-38-68-69-72-118-160-164-161-10-48-236-208$	1562544 (1.15%)	3121450(1.15%)	**84.32±2.94**

Table 7.10: Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the SUN397 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

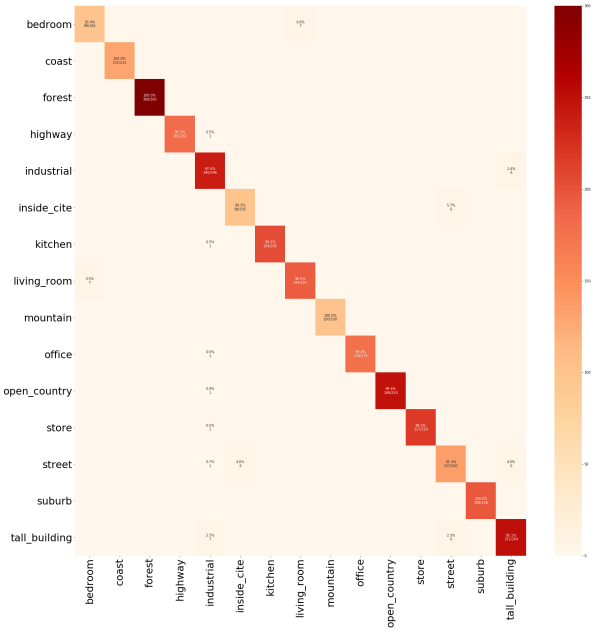

Figure 7.19: The confusion matrix of the first fold from the 5-fold cross-validation on the Scene15 dataset using the PulseNetOne pruned AlexNet model. It has an accuracy score of 97.99%, a precision score of 98.00%, recall score of 97.99% and a F1-score of 97.99%.

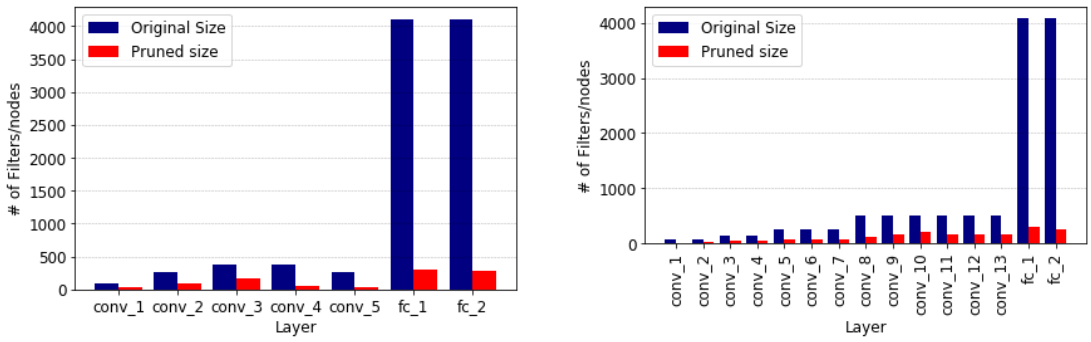

Figure 7.20: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the Scene15 dataset.

plex and expensive networks.

7.5.1 Discussion

PulseNetOne removes redundant filters in the convolutional layers (which helps to greatly improve inference timings) and nodes in the fully-connected layers (which helps to reduce storage cost). In this sense it can be considered a two-pronged attack on the network architecture, resulting in smaller and more efficient networks with better generalization largely caused (we believe) by a reduction in overfitting. To further investigate this belief, we would like to eval-

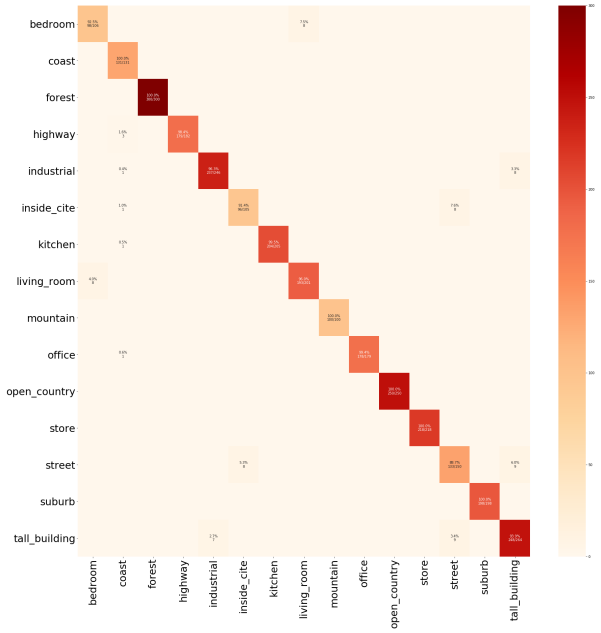

Figure 7.21: The confusion matrix of the first fold from the 5-fold cross-validation on the Scene15 dataset using the PulseNetOne pruned VGG16 model. It has an accuracy score of 97.46%, a precision score of 97.47%, recall score of 97.46% and a F1-score of 97.46%.

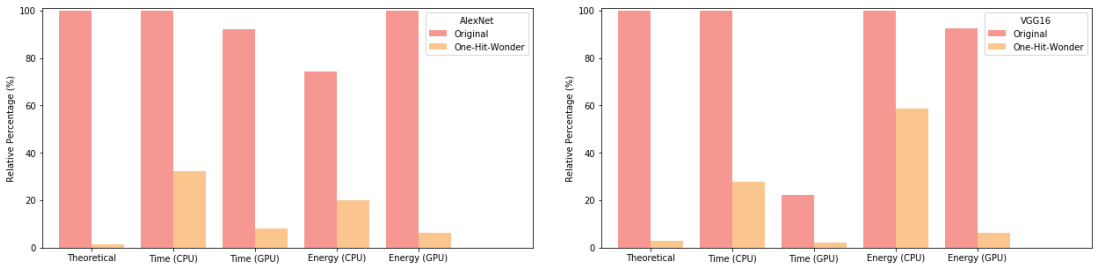

Figure 7.22: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the Scene15 dataset.

uate the pruned networks on different datasets, but from the same classification area to see if the inference networks generalized well. If the networks results in poor generalization, we would try a more comprehensive data augmentation. This helps the pruned networks to achieve state-of-the-art results in all the tested remote sensing benchmark datasets, at a fraction of the computational expense.

All the remote sensing and scene benchmark datasets used in this research were pruned in a similar fashion: the first and last few convolutional layers contain most redundant filters which are pruned, while the centre layers seem to carry more important details for the classification of the datasets (see Figures 7.7,

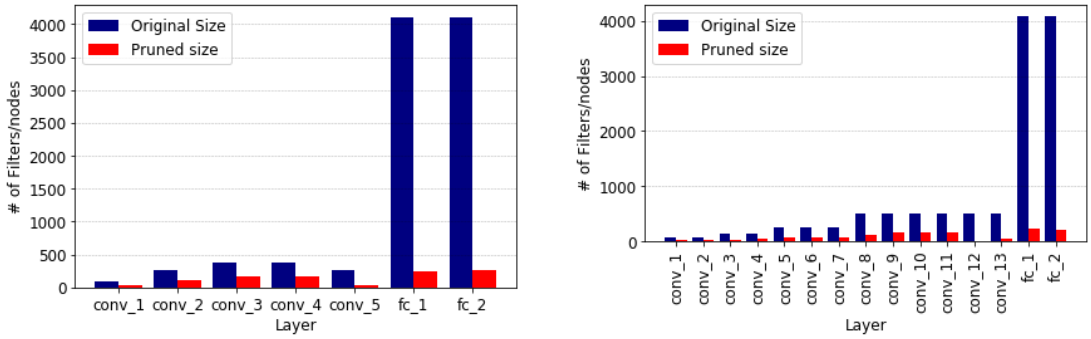

Figure 7.23: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the SUN397 dataset.

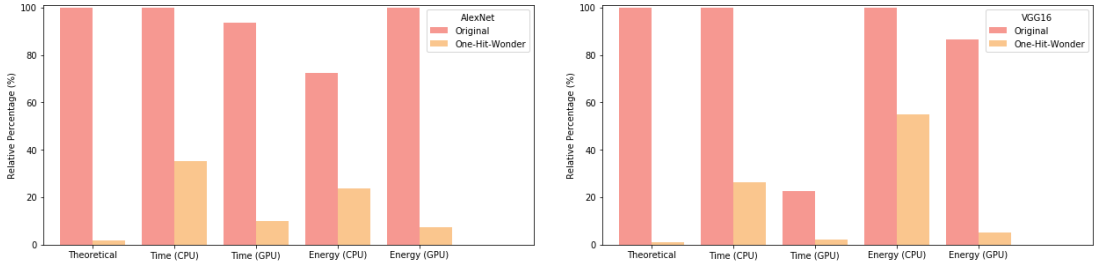

Figure 7.24: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the SUN397 dataset.

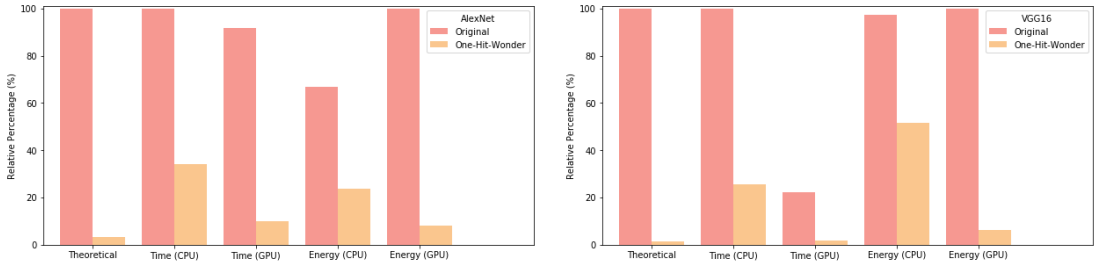

Figure 7.25: Bar-charts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the UC Merced dataset.

7.14, 7.16, 7.20, 7.23 and 7.28). In all the datasets the fully-connected layers were pruned significantly, confirming a theory in the literature that these layers are over-parameterized (hence some of the newer networks do not include them) [HPTD15]. We believe that, although they are over-parameterized, they can still add value if intelligently pruned.

The experiments had a faster inference time and consumed less energy, due to this, on the GPU compared to the CPU. But, in a real-world environment, it is unlikely that inference would be run on a GPU, rather on a CPU. Therefore,

Table 7.11: A comparison between state-of-the-art and PulseNetOne results on the SUN397 dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
AlexNet fine-tuned on Imagenet [ZLX+14]	2014	42.61
Otc and HOG [MZMT14]	2014	49.60
Hybrid-CNN [ZLX+14]	2014	53.86
AlexNet fine-tuned on Place205 [ZLX+14]	2014	54.32
GoogLeNet fine-tuned on Imagenet [SLJ+15]	2014	55.78
DSP [GWWL15]	2014	59.78
GoogLeNet fine-tuned on Place205 [SLJ+15]	2014	61.51
InterActive [XZW+16]	2016	62.97
G-MS2F (ADD) [TWK17]	2017	63.84
G-MS2F (Prod) [TWK17]	2017	64.06
Sparse Representation [NLB+17]	2017	71.08
MFAFVNet+Places [LDV17]	2017	72.01
MP [SJH17]	2017	72.60
SDO [CYY+18]	2018	73.41
Adi-Red [ZL18]	2018	73.59
SRG [ZC19]	2019	74.06
FOSNet CCG [SHK19]	2019	76.62
FOSNet CCM-CCG [SHK19]	2019	77.28
SOSF+CFA+GAF [SLL+18]	2018	78.93
AlexNet-PulseNetOne	2020	**82.11**
VGG16-PulseNetOne	2020	**84.32**

looking at the CPU experiments of both the original and PulseNetOne networks, it can be clearly seen that the pruned networks are significantly more efficient in all manners: storage, speed and energy.

Finally, from Tables 7.3, 7.4, 7.7, 7.8, 7.11 and 7.13 it can be seen that PulseNetOne obtains new state-of-the-art classification accuracy on all the remote sensing benchmark datasets. Our proposed method consistently outperforms current approaches by between 2% and 4% in most cases.

7.6 Conclusion

CNNs are state-of-the-art models for image classification, and although much success has been achieved using them in the remote sensing research area, our proposed method shows that the models being used as feature extractors, or as

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	$96-256-384-384-256-4096-4096$	58367381 (100%)	116715592(100%)	83.25±0.26
Transferred	-	-	-	94.42±0.18
Fine-tuned	-	-	-	98.19±0.24
PulseNetOne	$39-59-132-141-106-340-549$	1940812 (3.33%)	3878858(3.32%)	**99.82±0.11**
VGG16				
scratch	$64-64-128-128-256-256-256-512-512-512-512-512-4096-4096$	134346581 (100%)	268668304(100%)	75.01±0.32
Transferred	-	-	-	95.24±0.10
Fine-tuned	-	-	-	98.14±0.10
PulseNetOne	$10-20-31-29-70-79-83-185-160-165-172-13-55-272-288$	1886045 (1.40%)	3768800(1.40%)	**99.69±0.13**

Table 7.12: Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the UC Merced dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network retained is highlighted in red.

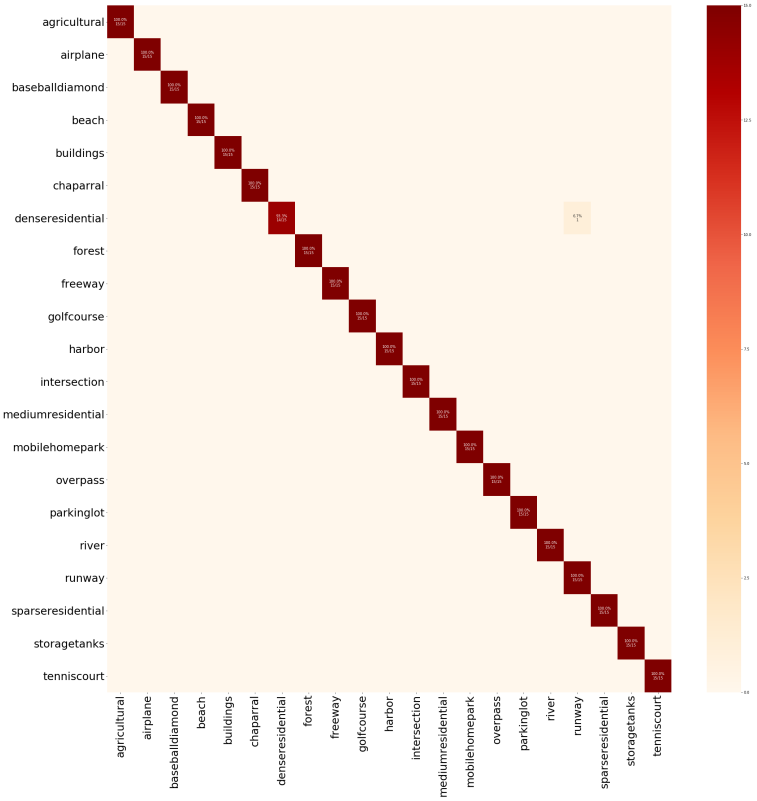

Figure 7.26: The confusion matrix of the first fold from the 5-fold cross-validation on the UC Merced dataset using the PulseNetOne pruned AlexNet model. It has an accuracy score of 99.68%, a precision score of 99.70%, recall score of 99.68% and a F1-score of 99.68%.

components of other techniques, are highly over-parameterized. We show that by pruning redundant filters and nodes, not only do we achieve better classification accuracy due to a strong regularization on the model, we also create a much more efficient network. PulseNetOne compresses AlexNet and VGG16 on average down to approximately 2% and 4% respectively of their original sizes. Its robustness is demonstrated using six remote sensing benchmark datasets, on which it greatly compresses the CNNs and achieves state-of-the-art classification accuracy. This was our final work done on the pruning of CNNs, and for the last chapter of the thesis, we will give the general conclusions of each chapter.

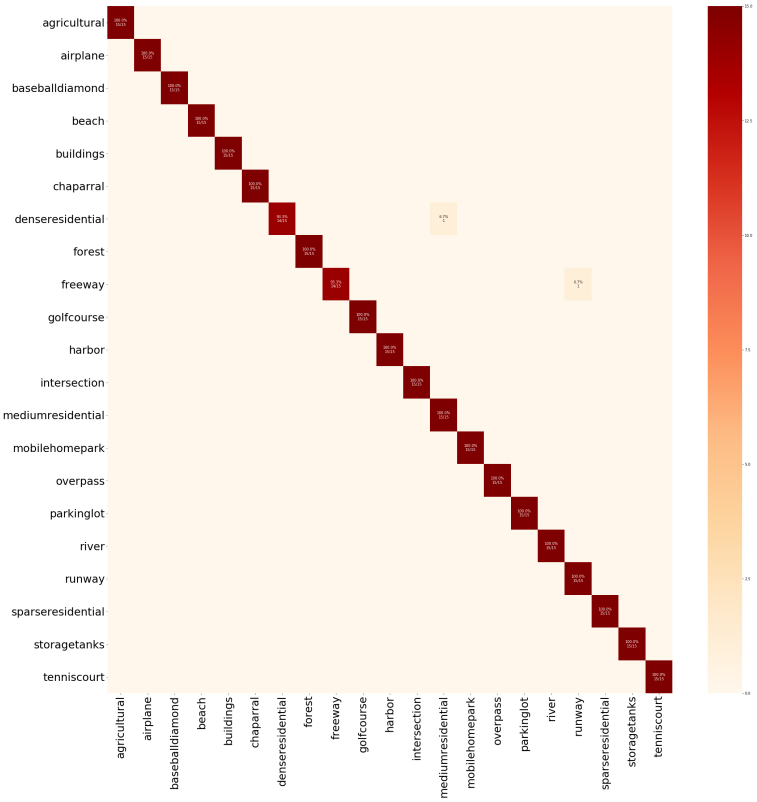

Figure 7.27: The confusion matrix of the first fold from the 5-fold cross-validation on the UC Merced dataset using the PulseNetOne pruned VGG16 model. It has an accuracy score of 99.37%, a precision score of 99.40%, recall score of 99.37% and a F1-score of 99.36%.

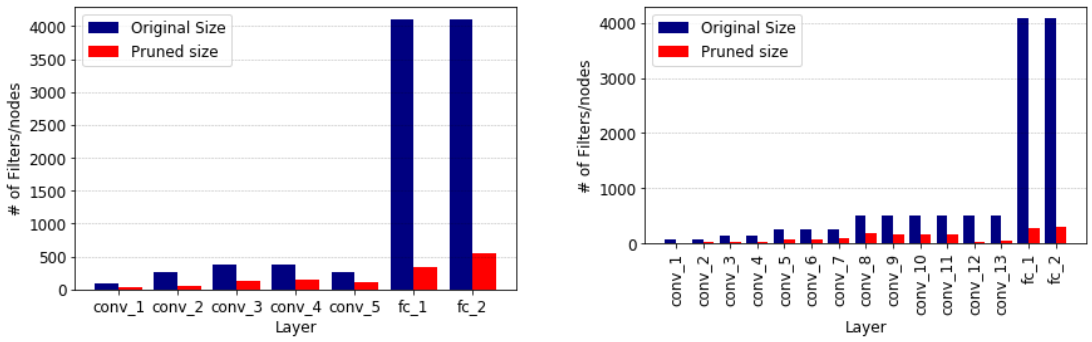

Figure 7.28: Comparison of layers between original and pruned version of both AlexNet and VGG16 on the UC Merced dataset.

Table 7.13: A comparison between state-of-the-art and PulseNetOne results on the UCM dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
TF-CNN [ZFZ16]	2016	89.90
Multiview deep learning [LSVdBM15]	2015	93.48
GBRCN [ZDZ15]	2015	94.53
CNN with OverFeat [MDES15]	2015	95.48
LGF [ZLCD16]	2016	95.48
Pre-trained AlexNet SPP [HZCZ17]	2017	95.95
Pre-trained AlexNet SPP-SS [HZCZ17]	2017	96.67
Discrimative CNNs AlexNet [CYY$^+$18]	2018	96.67
Discrimative CNNs GoogLeNet [CYY$^+$18]	2018	97.07
Fusion by addition [CLGY17]	2017	97.42
Pre-trained AlexNet 1st-FC [HXHZ15]	2015	98.49
VGG16-CapsNet [ZTZ19]	2019	98.81
Discrimative CNNs VGG16 [CYY$^+$18]	2018	98.93
GCFs+LOFs [CZT$^+$18]	2018	99.00
Inception-v3-CapsNet [ZTZ19]	2019	99.05
Fine-tuned GoogLeNet with SVM [NPdS17]	2017	99.47
VGG16-PulseNetOne	2020	**99.69**
AlexNet-PulseNetOne	2020	**99.82**

				Original		PulseNetOne		
Dataset	CNN	Processor	Storage (MB)	Energy per image (mJ)	Time per image (ms)	Storage (MB)	Energy (mJ)	Time per image (ms)
AID	AlexNet	CPU	222.795	1.47	46.16	5.89	0.51	15.89
-	-	GPU	-	2.68	43.46	-	0.19	4.44
-	VGG16	CPU	512.632	6.59	362.77	26.484	4.89	165.82
-	-	GPU	-	5.88	81.26	-	0.88	10.30
MIT67	AlexNet	CPU	223.373	1.67	46.07	5.105	0.16	17.11
-	-	GPU	-	2.42	42.79	-	0.14	4.10
-	VGG16	CPU	513.21	5.85	364.15	42.388	4.70	109.19
-	-	GPU	-	5.83	81.51	-	1.02	12.28
NWPU-RESISC45	AlexNet	CPU	223.029	1.69	45.78	5.105	0.35	13.24
-	-	GPU	-	2.48	42.37	-	0.15	4.07
-	VGG16	CPU	512.867	5.05	362.75	15.541	5.02	127.78
-	-	GPU	-	5.34	81.49	-	0.55	8.71
Scene15	AlexNet	CPU	222.56	1.66	45.79	2.899	0.45	14.76
-	-	GPU	-	2.24	42.24	-	0.14	3.65
-	VGG16	CPU	512.398	6.54	362.50	15.562	3.84	101.28
-	-	GPU	-	6.05	81.19	-	0.42	7.98
SUN397	AlexNet	CPU	228.53	1.75	46.02	4.218	0.57	16.21
-	-	GPU	-	2.42	43.10	-	0.18	4.52
-	VGG16	CPU	518.368	6.54	363.69	5.961	3.59	96.20
-	-	GPU	-	5.66	81.81	-	0.33	7.41
UC Merced	AlexNet	CPU	222.654	1.70	45.92	7.404	0.60	15.62
-	-	GPU	-	2.55	42.03	-	0.21	4.57
-	VGG16	CPU	512.492	5.79	362.45	7.195	3.06	92.31
-	-	GPU	-	5.95	80.41	-	0.38	6.65

Table 7.14: Computational results for datasets AID, MIT67, NWPU-RESISC45, Scene15, SUN397 and UC Merced using AlexNet and VGG16. It shows the storage space each network requires, the inference speed per image and the energy consumed per image of both the original and compressed networks using a batch size of a single image.

Chapter 8

Conclusions & Future Work

In this thesis, we have addressed the question of *how to reduce the complexity of classification tasks*. First, we have proposed a filter feature selection approach, based on the idea of dominance, which had 3 distinct variations; Relevance-Redundancy Dominance (RRD), Fast Relevance-Redundancy Dominance (FRRD) and improved Relevance-Redundancy Dominance with a Dominant Genetic Algorithm (iRRD-GA), which we implemented to reduce the problem complexity through dimensionality reduction. Next, we proposed a deep learning approach to reduce model complexity through pruning, which also had 3 variations; Pulse-Net, unsupervised PulseNet and PulseNetOne.

Throughout this thesis, using different datasets dependent on the task, we found that our complexity reduction techniques selected the relevant information to solve the classification problem it was assigned to. The variants of RRD (Chapters 2, 3, 4), achieved either current, or better than current state-of-the-art results using a minimum subset of relevant and un-dominated features in the data. Our different deep learning methods (Chapters 5, 6, 7) significantly decreased model complexity, in cases beyond current state-of-the-art, while maintaining minimum accuracy loss. Our findings address the main goal of this thesis: "reducing complexity of classification tasks."

8.1 Conclusions

In this section, we summarize the main conclusions of this thesis.

8.1.1 Relevance-Redundancy Dominance

In Chapter 2, we have introduced a filter feature selection method, based on the idea of dominance, called Relevance-Redundancy Dominance (RRD). We carried out a range of experiments on credit scoring data, using similar experimental design as that in the literature, in order for direct comparisons of our proposed approach with the current state-of-the-art. We evaluated our method, again keeping in-line with related work, on 3 classifiers; logistic regression, random forest and naive Bayes, resulting in comparable classification accuracy. An advantage of our method is that, depending on the training data, it can take any relevant statistic. Another added bonus is RRD requires no user input thresholds, instead allowing the algorithm to decide which, and how many, features needed to complete the classification task.

8.1.2 Fast Relevance-Redundancy Dominance

In Chapter 3, we have presented a faster version of RRD, named Fast Relevance-Redundancy Dominance (FRRD), which was aimed at high dimensional data. To this aim we evaluated our method on microarray data, which as the added difficulty of having a very small number of samples. In this part of the work, we displayed FRRD's robustness using 5 classifiers; random forest and naive Bayes, Support Vector Machine (SVM), C4.5 Decision Tree and k-nearest neighbours (KNN). A disadvantage of FRRD, was that the user predefined the number of features to retain before checking for dominance within the final subset. While, its main advantage was the speed at which it could select a subset of relevant features, making it very suitable to high-dimensional data.

8.1.3 Improved Relevance-Redundancy Dominance with a Genetic Algorithm

In Chapter 4, we have proposed a hybrid filter-wrapper feature selection method called, improved Relevance-Redundancy Dominance with a Genetic Algorithm (iRRD-GA), which again focused on microarray data similarly to FRRD. Our dominance-based filter method is our core contribution, which selects a small subset of relevant, un-dominated features, by using 2 different statistical viewpoints; information theory and chi-squared. Then, following similar approaches in the field, we implement a wrapper feature selector; genetic algorithm, which we slightly tailor to suit our method. The combination of this hybrid approach

was extensively tested using 4 different types of cross-validation, and with 10 different classifiers. The extensive analysis had 2 goals; to be a benchmark publication for other work in the area to compare the majority of experimental design results with, and secondly to show that once the training data is reduced to a *good relevant un-dominated* subset, less emphasize is on the choice of the evaluation classifier.

8.1.4 Pulse-Net

In Chapter 5, we move our focus to deep learning, reducing model complexity in this area, and introduce our pruning approach called Pulse-Net. We evaluate our method on 3 convolutional neural networks (CNNs), using 3 image datasets. We adapt the idea of dropout to simulate parts of the network being switched off, and from this, we iteratively increase the amount of the network being turned-off until we reach a point we consider the network to be fully compressed. We then extract the relevant sections of the network and reload them onto a smaller, overall more efficient network for the inference evaluation stage. We use an absolute L_1-norm as the decision metric, deeming low value filters/nodes to be of little importance, and prune accordingly.

8.1.5 Unsupervised PulseNet

In Chapter 6, we continue our work on pruning methods using the same 3 CNNS, and datasets for comparison, but propose a k-means based unsupervised approach named unsupervised PulseNet. This approach has 2 main advantages over the previous version; *non-requirement of user input to select pruning rate* and, *reduced network is extract and reloaded onto smaller network at each pruning iteration resulting in faster training times and more control over the accuracy loss*. In comparison to current state-of-the-art in this area, unsupervised PulseNet achieves a better compression-accuracy ratio.

8.1.6 PulseNetOne

In Chapter 7, we introduce our newest version of PulseNet, which is more application focused, called PulseNetOne. It has the same advantages as the previous unsupervised PulseNet, with the added improvement of being extremely faster at extracting a *good* compressed efficient network. Because we want this version to be targeted towards application use, we evaluate 2 CNNs on 6 remote-sensing

and scene datasets. From related work research for this part of the thesis, we believe we are the first to introduce pruning in this field. Our proposed method not only resulted in an efficient model, but also increased the classification accuracy by reducing over-fitting the training data.

8.2 Future Work

Our contributions in this thesis have introduced new ways to reduce both problem and model complexities in the area of classification. In traditional classification tasks, we have focused on dimensionality reduction, showing how we can efficiently and accurately evaluate high-dimensionality micro-array data, but we believe our work can be further expanded. We also have shown our pruning methods reduce model complexity, but believe these to be the stepping stones of further expansion work. Finally, we believe our work on human activity recognition is also just the beginning of more diverse classification problems to solve efficiently. In this section, we try to give a brief overview of some of these possible extensions.

8.2.1 IRRD-GA on big data

We have shown impressive results using IRRD-GA on high-dimensional microarray data. But to further test our dominance idea, both using the filter feature selection to significantly reduce the initial dataset to a subset, and the dominant genetic algorithm to a minimum set of relevant un-dominated features, we believe experiments on big data would be a very interesting future work. As a twist in our method, we would like to investigate the parallelization and/or a map-reduce approach of iRRD-DGA while performing big data experiments.

8.2.2 PulseNet's possible further exploration work

For future work other types of network may be explored, and with an unrestricted time limit it would be very interesting to see what compression rate can be achieved using the dataset ImageNet. By default k-means was selected as the unsupervised clustering method, but other clustering methods could be tested with the aim of further reducing computational cost and network compression. Finally, extensive exploration on different computer vision tasks (such as object detection and segmentation) using Unsupervised PulseNet would be

of great interest, as it might significantly improve deep learning efficiency on these problems.

References

[AA05] Ahmed Al-Ani. Feature subset selection using ant colony optimization. *International journal of computational intelligence*, 2005.

[AAY17] Reza Abbasi-Asl and Bin Yu. Structural compression of convolutional neural networks based on greedy filter pruning. *arXiv preprint arXiv:1705.07356*, 2017.

[ABA15a] Hala Alshamlan, Ghada Badr, and Yousef Alohali. mrmr-abc: a hybrid gene selection algorithm for cancer classification using microarray gene expression profiling. *Biomed research international*, 2015, 2015.

[ABA15b] Hala M Alshamlan, Ghada H Badr, and Yousef A Alohali. Genetic bee colony (gbc) algorithm: A new gene selection method for microarray cancer classification. *Computational biology and chemistry*, 56:49–60, 2015.

[ABN+99] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, June 1999.

[AED+00] Ash A Alizadeh, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503, 2000.

[AG13] Mohammad Javad Abdi and Davar Giveki. Automatic detection of erythemato-squamous diseases using pso–svm based

on association rules. *Engineering Applications of Artificial Intelligence*, 26(1):603–608, 2013.

[Agr03] Alan Agresti. *Categorical data analysis*, volume 482. John Wiley & Sons, 2003.

[AHR12] Mohammad Javad Abdi, Seyed Mohammad Hosseini, and Mansoor Rezghi. A novel weighted support vector machine based on particle swarm optimization for gene selection and tumor classification. *Computational and Mathematical Methods in Medicine*, 2012, 2012.

[ALA16] Javier Apolloni, Guillermo Leguizamón, and Enrique Alba. Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments. *Applied Soft Computing*, 38:922–932, 2016.

[AN07a] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[AN07b] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[Ari] Arizona. Feature Selection Datasets at Arizona State University. `http://featureselection.asu.edu/datasets.php`. Accessed: 2016-07-12.

[ASS+02] Scott A Armstrong, Jane E Staunton, Lewis B Silverman, Rob Pieters, Monique L den Boer, Mark D Minden, Stephen E Sallan, Eric S Lander, Todd R Golub, and Stanley J Korsmeyer. Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature genetics*, 30(1):41–47, 2002.

[ATH03] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 577–584, 2003.

[BCAB18] Verónica Bolón-Canedo and Amparo Alonso-Betanzos. *Recent advances in ensembles for feature selection*, volume 147. Springer, 2018.

[BCD15] Reza Bahmanyar, Shiyong Cui, and Mihai Datcu. A comparative study of bag-of-words and bag-of-topics models of eo image patches. *IEEE Geoscience and Remote Sensing Letters*, 12(6):1357–1361, 2015.

[BCSMAB10] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. On the effectiveness of discretization on gene selection of microarray data. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[BCSMAB11] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. Toward an ensemble of filters for classification. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 331–336. IEEE, 2011.

[BCSMAB12] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. An ensemble of filters and classifiers for microarray data classification. *Pattern Recognition*, 45(1):531–539, 2012.

[BCSMAB14a] Verónica Bolón-Canedo, Noelia Sánchez-Marono, and Amparo Alonso-Betanzos. Data classification using an ensemble of filters. *Neurocomputing*, 135:13–20, 2014.

[BCSMAB+14b] Verónica Bolón-Canedo, Noelia Sánchez-Marono, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[BCSMAB+14c] Verónica Bolón-Canedo, Noelia Sánchez-Marono, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[BCSMAB15] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. Distributed feature selection: An application to microarray data classification. *Applied soft computing*, 30:136–150, 2015.

[BCTD17] Xiaoyong Bian, Chen Chen, Long Tian, and Qian Du. Fusing local and global features for high-resolution scene classifica-

tion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(6):2889–2901, 2017.

[Bel55] Richard Bellman. Transactions of the symposium on computing, mechanics, statistics, and partial differential equations, volume ii., 1955.

[BGP19] David Browne, Michael Giering, and Steven Prestwich. Pulsenet: Dynamic compression of convolutional neural networks. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 346–351. IEEE, 2019.

[BHDH+11] Edmundo Bonilla-Huerta, Béatrice Duval, José C Hernández Hernández, Jin-Kao Hao, and Roberto Morales-Caporal. Hybrid filter-wrapper with a specialized random multi-parent crossover operator for gene selection and classification problems. In *International Conference on Intelligent Computing*, pages 453–461. Springer, 2011.

[BKH+02] David G Beer, Sharon LR Kardia, Chiang-Ching Huang, Thomas J Giordano, Albert M Levin, David E Misek, Lin Lin, Guoan Chen, Tarek G Gharib, Dafydd G Thomas, et al. Gene-expression profiles predict survival of patients with lung adenocarcinoma. *Nature medicine*, 8(8):816–824, 2002.

[BM10] Gianluca Bontempi and Patrick E Meyer. Causal filter selection in microarray data. In *Proceedings of the 27th international conference on machine learning (icml-10)*, pages 95–102, 2010.

[BMP16] David Browne, Carlo Manna, and Steven D Prestwich. Relevance-redundancy dominance: a threshold-free approach to filter-based feature selection. In *24th Irish Conference on Artificial Intelligence and Cognitive Science 2016*. Sun SITE Central Europe/RWTH Aachen University, 2016.

[Bre17] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[BRS+01] Arindam Bhattacharjee, William G Richards, Jane Staunton, Cheng Li, Stefano Monti, Priya Vasa, Christine Ladd, Javad Beheshti, Raphael Bueno, Michael Gillette, et al. Classification of human lung carcinomas by mrna expression profiling

reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences*, 98(24):13790–13795, 2001.

[BSZ17] Mehrab Ghanat Bari, Sirajul Salekin, and Jianqiu Zhang. A robust and efficient feature selection algorithm for microarray data. *Molecular informatics*, 36(4):1600099, 2017.

[CBD15] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[CDF+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[CH03] Mu-Chen Chen and Shih-Hsien Huang. Credit scoring and rejected instances reassigning through evolutionary computation techniques. *Expert Systems with Applications*, 24(4):433–441, 2003.

[Cha09] Sounak Chakraborty. Bayesian binary kernel probit model for microarray based cancer classification and gene selection. *Computational Statistics & Data Analysis*, 53(12):4198–4209, 2009.

[Che13] Anil M Cheriyadat. Unsupervised feature learning for aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(1):439–451, 2013.

[CHL17] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[CHS+16] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[CHYW09] Ruichu Cai, Zhifeng Hao, Xiaowei Yang, and Wen Wen. An efficient gene selection algorithm based on mutual information. *Neurocomputing*, 72(4-6):991–999, 2009.

[CKY16] Li-Yeh Chuang, Chao-Hsuan Ke, and Cheng-Hong Yang. A hybrid both filter and wrapper feature selection method for microarray classification. *arXiv preprint arXiv:1612.08669*, 2016.

[CL10] Fei-Long Chen and Feng-Chia Li. Combination of feature selection approaches with svm in credit scoring. *Expert systems with applications*, 37(7):4902–4909, 2010.

[CLG+04] Sabina Chiaretti, Xiaochun Li, Robert Gentleman, Antonella Vitale, Marco Vignetti, Franco Mandelli, Jerome Ritz, and Robin Foa. Gene expression profile of adult t-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, 103(7):2771–2778, 2004.

[CLGY17] Souleyman Chaib, Huan Liu, Yanfeng Gu, and Hongxun Yao. Deep feature fusion for vhr remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(8):4775–4784, 2017.

[CN09] Ying Chen and Danh V Nguyen. Identification of relevant genes from microarray experiments based on partial least squares weights: application to cancer genomics. In *Computational Biology*, pages 1–17. Springer, 2009.

[CTSO18] Elliot J Crowley, Jack Turner, Amos Storkey, and Michael O'Boyle. Pruning neural networks: is it time to nip it in the bud? *arXiv preprint arXiv:1810.04622*, 2018.

[CWT+15] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294, 2015.

[CYWY11] Li-Yeh Chuang, Cheng-Huei Yang, Kuo-Chuan Wu, and Cheng-Hong Yang. A hybrid feature selection method for dna microarray data. *Computers in biology and medicine*, 41(4):228–237, 2011.

[CYXH12] Xiaojun Chen, Yunming Ye, Xiaofei Xu, and Joshua Zhexue Huang. A feature group weighting method for subspace

clustering of high-dimensional data. *Pattern Recognition*, 45(1):434–446, 2012.

[CYY+18] Gong Cheng, Ceyuan Yang, Xiwen Yao, Lei Guo, and Junwei Han. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns. *IEEE transactions on geoscience and remote sensing*, 56(5):2811–2821, 2018.

[CZS+16] Chen Chen, Baochang Zhang, Hongjun Su, Wei Li, and Lu Wang. Land-use scene classification using multi-scale completed local binary patterns. *Signal, image and video processing*, 10(4):745–752, 2016.

[CZT+18] Guanzhou Chen, Xiaodong Zhang, Xiaoliang Tan, Yufeng Cheng, Fan Dai, Kun Zhu, Yuanfu Gong, and Qing Wang. Training small networks for scene classification of remote sensing images via knowledge distillation. *Remote Sensing*, 10(5):719, 2018.

[CZY+16] Gong Cheng, Peicheng Zhou, Xiwen Yao, Chao Yao, Yanbang Zhang, and Junwei Han. Object detection in vhr optical remote sensing images via learning rotation-invariant hog feature. In *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, pages 433–436. IEEE, 2016.

[Das17] Rasmita Dash. A two stage grading approach for feature selection and classification of microarray data using pareto based feature ranking techniques: A case study. *Journal of King Saud University-Computer and Information Sciences*, 2017.

[DB03] Marcel Dettling and Peter Bühlmann. Boosting for tumor classification with gene expression data. *Bioinformatics*, 19(9):1061–1069, 2003.

[DB17] M Dashtban and Mohammadali Balafar. Gene selection for microarray cancer classification using a new evolutionary method employing artificial intelligence concepts. *Genomics*, 109(2):91–107, 2017.

[DCP17] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 4857–4867, 2017.

[DDHT18] Xiaohan Ding, Guiguang Ding, Jungong Han, and Sheng Tang. Auto-balanced filter pruning for efficient convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[DKS95] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.

[DP05] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(02):185–205, 2005.

[DPHC18] Asit Kumar Das, Soumen Kumar Pati, Hsien-Hung Huang, and Chi-Ken Chen. Cancer classification by gene subset selection from microarray dataset. *J. UCS*, 24(6):682–710, 2018.

[DR03] Kalyanmoy Deb and A Raji Reddy. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, 72(1-2):111–129, 2003.

[DSD+13] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[dSPdST10] Jefersson Alex dos Santos, Otávio Augusto Bizetto Penatti, and Ricardo da Silva Torres. Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification. In *VISAPP (2)*, pages 203–208, 2010.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[FCVF+04] William A Freije, F Edmundo Castro-Vargas, Zixing Fang, Steve Horvath, Timothy Cloughesy, Linda M Liau, Paul S Mischel, and Stanley F Nelson. Gene expression profiling of gliomas strongly predicts survival. *Cancer research*, 64(18):6503–6510, 2004.

[FD81] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[FS+96] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[GB13] Fernando González and Lluís A Belanche. Feature selection for microarray gene expression data using simulated annealing guided by the multivariate joint entropy. *arXiv preprint arXiv:1302.1733*, 2013.

[GBGK12] Enrico Glaab, Jaume Bacardit, Jonathan M Garibaldi, and Natalio Krasnogor. Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data. *PloS one*, 7(7), 2012.

[GBS+19] Manosij Ghosh, Shemim Begum, Ram Sarkar, Debasis Chakraborty, and Ujjwal Maulik. Recursive memetic algorithm for gene selection in microarray data. *Expert Systems with Applications*, 116:172–185, 2019.

[GGNZ08] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[GHT14] John Q Gan, Bashar Awwad Shiekh Hasan, and Chun Sing Louis Tsui. A filter-dominating hybrid sequential forward

floating search method for feature subset selection in high-dimensional space. *International Journal of Machine Learning and Cybernetics*, 5(3):413–423, 2014.

[GJH+02] Gavin J Gordon, Roderick V Jensen, Li-Li Hsiao, Steven R Gullans, Joshua E Blumenstock, Sridhar Ramaswamy, William G Richards, David J Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer research*, 62(17):4963–4967, 2002.

[GK79] Leo A Goodman and William H Kruskal. Measures of association for cross classifications. In *Measures of association for cross classifications*, pages 2–34. Springer, 1979.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[GST+99] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[GWGL14] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[GWWL15] Bin-Bin Gao, Xiu-Shen Wei, Jianxin Wu, and Weiyao Lin. Deep spatial pyramid: The devil is once again in the details. *arXiv preprint arXiv:1504.05277*, 2015.

[GYC16] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, pages 1379–1387, 2016.

[GZL+16] Ruiquan Ge, Manli Zhou, Youxi Luo, Qinghan Meng, Guoqin Mai, Dongli Ma, Guoqing Wang, and Fengfeng Zhou. Mctwo:

a two-step feature selection algorithm based on maximal information coefficient. *BMC bioinformatics*, 17(1):142, 2016.

[Hal99a] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[Hal99b] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[HB02] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):289–300, 2002.

[HC06] Jin-Hyuk Hong and Sung-Bae Cho. Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognition Letters*, 27(2):143–150, 2006.

[HC07] Hui-Ling Huang and Fang-Lin Chang. Esvm: Evolutionary support vector machine for automatic feature selection and classification of microarray data. *Biosystems*, 90(2):516–528, 2007.

[HCLD16] Longhui Huang, Chen Chen, Wei Li, and Qian Du. Remote sensing image scene classification using multi-scale completed local binary patterns and fisher vectors. *Remote Sensing*, 8(6):483, 2016.

[HCW07] Cheng-Lung Huang, Mu-Chen Chen, and Chieh-Jen Wang. Credit scoring with a data mining approach based on support vector machines. *Expert systems with applications*, 33(4):847–856, 2007.

[HCX07] Jinjie Huang, Yunze Cai, and Xiaoming Xu. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters*, 28(13):1825–1844, 2007.

[HDH06] Edmundo Bonilla Huerta, Béatrice Duval, and Jin-Kao Hao. A hybrid ga/svm approach for gene selection and classification of microarray data. In *Workshops on Applications of Evolutionary Computation*, pages 34–44. Springer, 2006.

[HGC08] D Huang, Zhaohui Gan, and Tommy WS Chow. Enhanced feature selection models using gradient-based and point injection techniques. *Neurocomputing*, 71(16-18):3114–3123, 2008.

[HJL16] Luis Herranz, Shuqiang Jiang, and Xiangyang Li. Scene recognition with cnns: objects, scales and dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 571–579, 2016.

[HKD+18] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[HLC+11] Bin Han, Lihua Li, Yan Chen, Lei Zhu, and Qi Dai. A two step method to identify clinical outcome relevant genes with microarray data. *Journal of biomedical informatics*, 44(2):229–238, 2011.

[HLF18] Dongmei Han, Qigang Liu, and Weiguo Fan. A new image classification method using cnn transfer learning and web data augmentation. *Expert Systems with Applications*, 95:43–56, 2018.

[HLH07] Hui-Ling Huang, Chong-Cheng Lee, and Shinn-Ying Ho. Selecting a minimal number of relevant genes from microarray data to design accurate tissue classifiers. *Biosystems*, 90(1):78–86, 2007.

[HLM+16] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[HLW+19] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[HMD15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning,

trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[HS99] MA Hall and LA Smith. Feature subset selection: A correlation based filter approach. In *Proc. of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, pages 855–858, 1999.

[HSK+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HXHZ15] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680–14707, 2015.

[HZC+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[HZCZ17] Xiaobing Han, Yanfei Zhong, Liqin Cao, and Liangpei Zhang. Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. *Remote Sensing*, 9(8):848, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE*

international conference on computer vision, pages 1026–1034, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[HZS17] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[HZYN18] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018.

[IHM+16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[Jin14] Si-Yuan Jing. A hybrid genetic algorithm for feature subset selection in rough set theory. *Soft Computing*, 18(7):1373–1382, 2014.

[JKP94] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994.

[JL12] David Jin and Sally Lin. *Advances in mechanical and electronic engineering*. Springer, 2012.

[JW16] Sheng-yi Jiang and Lian-xi Wang. Efficient feature selection based on correlation measure between continuous and discrete features. *Information Processing Letters*, 116(2):203–215, 2016.

[KAAAJ11] Rami N Khushaba, Ahmed Al-Ani, and Adel Al-Jumaily. Feature subset selection using differential evolution and a statistical repair mechanism. *Expert Systems with Applications*, 38(9):11515–11526, 2011.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[KŁ16] Jerzy Krawczuk and Tomasz Łukaszuk. The feature selection bias problem in relation to high-dimensional gene data. *Artificial intelligence in medicine*, 66:63–71, 2016.

[KM13] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[KMW11] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *2011 International Conference on Digital Image Computing: Techniques and Applications*, pages 501–506. IEEE, 2011.

[Kon94] Igor Kononenko. Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[KPH07] Alexandros Kalousis, Julien Prados, and Melanie Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12(1):95–116, 2007.

[KPY+15] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[KR92] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.

[KRMV15] P Ganesh Kumar, Chellasamy Rani, D Mahibha, and T Aruldoss Albert Victoire. Fuzzy-rough-neural-based f-information for gene selection and sample classification. *International Journal of Data Mining and Bioinformatics*, 11(1):31–52, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks.

In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KSM15] Subhajit Kar, Kaushik Das Sharma, and Madhubanti Maitra. Gene selection from microarray gene expression data for classification of cancer subgroups employing pso and adaptive k-nearest neighborhood technique. *Expert Systems with Applications*, 42(1):612–627, 2015.

[KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[KWR+01] Javed Khan, Jun S Wei, Markus Ringner, Lao H Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R Antonescu, Carsten Peterson, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673, 2001.

[LAT18] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[LCL+05] Jane Jijun Liu, Gene Cutler, Wuxiong Li, Zheng Pan, Sihua Peng, Tim Hoey, Liangbiao Chen, and Xuefeng Bruce Ling. Multiclass cancer classification and biomarker discovery using ga-based algorithms. *Bioinformatics*, 21(11):2691–2697, 2005.

[LCT02] Yi Li, Colin Campbell, and Michael Tipping. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18(10):1332–1339, 2002.

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[LCY+17] Huijuan Lu, Junying Chen, Ke Yan, Qun Jin, Yu Xue, and Zhigang Gao. A hybrid feature selection algorithm for gene expression data classification. *Neurocomputing*, 256:56–62, 2017.

[LDV17] Yunsheng Li, Mandar Dixit, and Nuno Vasconcelos. Deep scene image classification with the mfafvnet. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5746–5754, 2017.

[LH08] Yukyee Leung and Yeungsam Hung. A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(1):108–117, 2008.

[LH17] Yishu Liu and Chao Huang. Scene classification via triplet networks. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(1):220–237, 2017.

[LHSL17] Qingshan Liu, Renlong Hang, Huihui Song, and Zhi Li. Learning multiscale deep features for high-resolution satellite image scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(1):117–126, 2017.

[LJL+14] Bo Liao, Yan Jiang, Wei Liang, Wen Zhu, Lijun Cai, and Zhi Cao. Gene selection using locality sensitive laplacian score. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(6):1146–1156, 2014.

[LKD+16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[LL02] Jinyan Li and Huiqing Liu. Kent ridge bio-medical data set repository. *Institute for Infocomm Research. http://sdmc. lit. org. sg/GEDatasets/Datasets. html*, 2002.

[LL11] Chien-Pang Lee and Yungho Leu. A novel hybrid feature selection method for microarray data analysis. *Applied Soft Computing*, 11(1):208–213, 2011.

[LLCK11] Chien-Pang Lee, Wen-Shin Lin, Yuh-Min Chen, and Bo-Jein Kuo. Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbor method. *Expert Systems with Applications*, 38(5):4661–4667, 2011.

[LLM14] Xiaoqiang Lu, Xuelong Li, and Lichao Mou. Semi-supervised multitask learning for scene recognition. *IEEE transactions on cybernetics*, 45(9):1967–1976, 2014.

[LLS+17] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[LO10] Qiang Lou and Zoran Obradovic. Feature selection by approximating the markov blanket in a kernel-induced space. In *ECAI*, pages 797–802, 2010.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LRvVW06] Carmen Lai, Marcel JT Reinders, Laura J van't Veer, and Lodewyk FA Wessels. A comparison of univariate and multivariate gene selection techniques for classification of cancer datasets. *BMC bioinformatics*, 7(1):235, 2006.

[LS+96] Huan Liu, Rudy Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[LSVdBM15] Francois PS Luus, Brian P Salmon, Frans Van den Bergh, and Bodhaswar Tikanath Jugpershad Maharaj. Multiview deep learning for land-use classification. *IEEE Geoscience and Remote Sensing Letters*, 12(12):2448–2452, 2015.

[LSZ+18] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[LWF+15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[LWH08] Shutao Li, Xixian Wu, and Xiaoyan Hu. Gene selection using genetic algorithm and support vectors machines. *Soft computing*, 12(7):693–698, 2008.

[LWL17] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[LWT08] Shutao Li, Xixian Wu, and Mingkui Tan. Gene selection using hybrid particle swarm optimization and genetic algorithm. *Soft Computing*, 12(11):1039–1048, 2008.

[LXM+18] Bao-Di Liu, Wen-Yang Xie, Jie Meng, Ye Li, and Yanjiang Wang. Hybrid collaborative representation for remote-sensing image scene classification. *Remote Sensing*, 10(12):1934, 2018.

[LZL+16] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, and Xiaoou Tang. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[LZO04] Tao Li, Chengliang Zhang, and Mitsunori Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.

[MAV17] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[MDES15] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2015.

[MDI05] Mohd Saberi Mohamad, Safaai Deris, and Rosli Md Illias. A hybrid of genetic algorithm and support vector machine for features selection and classification of gene expression microarray. *International Journal of Computational Intelligence and Applications*, 5(01):91–107, 2005.

[MLDD17] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 571–582. IEEE, 2017.

[MM72] Robert Messenger and Lewis Mandell. A modal search technique for predictive nominal scale multivariate analysis. *Journal of the American statistical association*, 67(340):768–772, 1972.

[MM07] Pritha Mahata and Kaushik Mahata. Selecting differentially expressed genes using minimum probability of classification error. *Journal of Biomedical Informatics*, 40(6):775–786, 2007.

[MM13] Monalisa Mandal and Anirban Mukhopadhyay. An improved minimum redundancy maximum relevance approach for feature selection in gene expression data. *Procedia Technology*, 10(0):20–27, 2013.

[MM16] Maryam Mollaee and Mohammad Hossein Moattar. A novel feature extraction approach based on ensemble feature selection and modified discriminant independent component analysis for microarray data classification. *Biocybernetics and Biomedical Engineering*, 36(3):521–529, 2016.

[MNSMN17] Habib Motieghader, Ali Najafi, Balal Sadeghi, and Ali Masoudi-Nejad. A hybrid gene selection algorithm for microarray cancer classification using genetic algorithm and learning automata. *Informatics in Medicine Unlocked*, 9:246–254, 2017.

[MODY11] Mohd Saberi Mohamad, Sigeru Omatu, Safaai Deris, and Michifumi Yoshioka. A modified binary particle swarm optimization for selecting the small subset of informative genes

from gene expression data. *IEEE Transactions on information Technology in Biomedicine*, 15(6):813–822, 2011.

[MR09] Piyushkumar A Mundra and Jagath C Rajapakse. Svm-rfe with mrmr filter for gene selection. *IEEE transactions on nanobioscience*, 9(1):31–37, 2009.

[MS11] Debahuti Mishra and Barnali Sahu. Feature selection for cancer classification: a signal-to-noise ratio approach. *International Journal of Scientific & Engineering Research*, 2(4):1–7, 2011.

[MT00] Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In *Advances in neural information processing systems*, pages 505–511, 2000.

[MTK+16] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[MWF14] Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information Sciences*, 286:228–246, 2014.

[MZMT14] Ran Margolin, Lihi Zelnik-Manor, and Ayellet Tal. Otc: A novel local descriptor for scene classification. In *European Conference on Computer Vision*, pages 377–391. Springer, 2014.

[MZO17] Nur Shazila Mohamed, Suhaila Zainudin, and Zulaiha Ali Othman. Metaheuristic approach for an enhanced mrmr filter method for classification using drug response microarray data. *Expert Systems with Applications*, 90:224–231, 2017.

[NF77] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, (9):917–922, 1977.

[NLB+17] Guilherme Nascimento, Camila Laranjeira, Vinicius Braz, Anisio Lacerda, and Erickson R Nascimento. A robust indoor scene recognition method based on sparse representation. In

Iberoamerican Congress on Pattern Recognition, pages 408–415. Springer, 2017.

[NM09a] Félix F González Navarro and Lluís A Belanche Muñoz. Gene subset selection in microarray data using entropic filtering for cancer classification. *Expert Systems*, 26(1):113–124, 2009.

[NM09b] Félix F González Navarro and Lluís A Belanche Muñoz. Gene subset selection in microarray data using entropic filtering for cancer classification. *Expert Systems*, 26(1):113–124, 2009.

[NPdS17] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.

[OGO+01] Paul O'Dea, Josephine Griffith, Colm O'Riordan, Josephine Griffith, and Colm O Riordan. Combining feature selection and neural networks for solving classification problems. *Information Technology Department, National University of Ireland*, 2001.

[OHT05] Chorng-Shyong Ong, Jih-Jeng Huang, and Gwo-Hshiung Tzeng. Building credit scoring models using genetic programming. *Expert Systems with Applications*, 29(1):41–47, 2005.

[OLM04] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Hybrid genetic algorithms for feature selection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11):1424–1437, 2004.

[PIAH+02] Emanuel F Petricoin III, Ali M Ardekani, Ben A Hitt, Peter J Levine, Vincent A Fusaro, Seth M Steinberg, Gordon B Mills, Charles Simone, David A Fishman, Elise C Kohn, et al. Use of proteomic patterns in serum to identify ovarian cancer. *The lancet*, 359(9306):572–577, 2002.

[PLD05a] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[PLD05b] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[PNDS15] Otávio AB Penatti, Keiller Nogueira, and Jefersson A Dos Santos. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–51, 2015.

[PTG+02] Scott L Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M Sturla, Michael Angelo, Margaret E McLaughlin, John YH Kim, Liliana C Goumnerova, Peter M Black, Ching Lau, et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[PTVF88] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c, 1988.

[PW15] Adam Polyak and Lior Wolf. Channel-level acceleration of deep face representations. *IEEE Access*, 3:2163–2175, 2015.

[QT09] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

[RBK+14] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[RG02] Chotirat Ann Ratanamahatana and Dimitrios Gunopulos. Scaling up the naive bayesian classifier: Using decision trees for feature selection. 2002.

[RORF16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[Ros61] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[SAEF17] Hanaa Salem, Gamal Attiya, and Nawal El-Fishawy. Classification of human cancer diseases by gene expression profiles. *Applied Soft Computing*, 50:124–134, 2017.

[SANA12] Salam Salameh Shreem, Salwani Abdullah, Mohd Zakree Ahmad Nazri, and Malek Alzaqebah. Hybridizing relieff, mrmr filters and ga wrapper approaches for gene selection. *J. Theor. Appl. Inf. Technol*, 46(2):1034–1039, 2012.

[SBS+07] Avrum Spira, Jennifer E Beane, Vishal Shah, Katrina Steiling, Gang Liu, Frank Schembri, Sean Gilman, Yves-Martine Dumas, Paul Calner, Paola Sebastiani, et al. Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. *Nature medicine*, 13(3):361–366, 2007.

[SFR+02] Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209, 2002.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[SHK19] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Fosnet: An end-to-end trainable deep neural network for scene recognition. *arXiv preprint arXiv:1907.07570*, 2019.

[SIM11] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):754–764, 2011.

[SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[SJH17] Xinhang Song, Shuqiang Jiang, and Luis Herranz. Multi-scale multi-feature context modeling for scene recognition in the semantic manifold. *IEEE Transactions on Image Processing*, 26(6):2721–2735, 2017.

[SK03] Shirish Krishnaj Shevade and S Sathiya Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

[SK07] Shital Shah and Andrew Kusiak. Cancer gene search with data-mining and genetic algorithms. *Computers in biology and medicine*, 37(2):251–261, 2007.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SLL+18] Ning Sun, Wenli Li, Jixin Liu, Guang Han, and Cong Wu. Fusing object semantics and deep appearance features for scene recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(6):1715–1728, 2018.

[SM02] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[SM12] Barnali Sahu and Debahuti Mishra. A novel feature selection algorithm using particle swarm optimization for cancer microarray data. *Procedia Engineering*, 38:27–31, 2012.

[SMM16] Fatemeh Vafaee Sharbaf, Sara Mosafer, and Mohammad Hossein Moattar. A hybrid gene selection approach for microarray data classification using cellular learning automata and ant colony optimization. *Genomics*, 107(6):231–238, 2016.

[SNW11] Qinbao Song, Jingjie Ni, and Guangtao Wang. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, 25(1):1–14, 2011.

[SPBCAB16] Borja Seijo-Pardo, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Using a feature selection ensemble on dna microarray datasets. In *ESANN*, 2016.

[Spe61] Charles Spearman. The proof and measurement of association between two things. 1961.

[SRT+02] Margaret A Shipp, Ken N Ross, Pablo Tamayo, Andrew P Weng, Jeffery L Kutok, Ricardo CT Aguiar, Michelle Gaasenbeek, Michael Angelo, Michael Reich, Geraldine S Pinkus, et al. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine*, 8(1):68–74, 2002.

[SSKY07] Qi Shen, Wei-Min Shi, Wei Kong, and Bao-Xian Ye. A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification. *Talanta*, 71(4):1679–1683, 2007.

[SWT16] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Sparsifying neural network connections for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4856–4864, 2016.

[SYXS12] Guofeng Sheng, Wen Yang, Tao Xu, and Hong Sun. High-resolution satellite scene classification using a sparse coding based multiple feature combination. *International journal of remote sensing*, 33(8):2395–2412, 2012.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[SZL+05] Quan-Sen Sun, Sheng-Gen Zeng, Yan Liu, Pheng-Ann Heng, and De-Shen Xia. A new method of feature fusion and its application in image recognition. *Pattern Recognition*, 38(12):2437–2448, 2005.

[TDE18] Siyabend Turgut, Mustafa Dağtekin, and Tolga Ensari. Microarray breast cancer data classification using machine learning methods. In *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, pages 1–3. IEEE, 2018.

[TES17] Sara Tarek, Reda Abd Elwahab, and Mahmoud Shoman. Gene expression based cancer classification. *Egyptian Informatics Journal*, 18(3):151–159, 2017.

[The72] Henry Theil. Statistical decomposition analysis; with applications in the social and administrative sciences. Technical report, 1972.

[TJGNA08] El-Ghazali Talbi, Laetitia Jourdan, José Garcia-Nieto, and Enrique Alba. Comparison of population based metaheuristics for feature selection: Application to microarray data classification. In *2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 45–52. IEEE, 2008.

[TK39] Aleksandr Aleksandrovich Tschuprow and M Kantorowitcsch. Principles of the mathematical theory of correlation. Technical report, William Hodge New York, 1939.

[TM18] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[TNRM15] Sina Tabakhi, Ali Najafi, Reza Ranjbar, and Parham Moradi. Gene selection for microarray data classification using a novel ant colony optimization. *Neurocomputing*, 168:1024–1036, 2015.

[TWK17] Pengjie Tang, Hanli Wang, and Sam Kwong. G-ms2f: Googlenet based multi-stage feature fusion of deep cnn for scene recognition. *Neurocomputing*, 225:188–197, 2017.

[TZW+03] Erming Tian, Fenghuang Zhan, Ronald Walker, Erik Rasmussen, Yupo Ma, Bart Barlogie, and John D Shaughnessy Jr. The role of the wnt-signaling antagonist dkk1 in the develop-

ment of osteolytic lesions in multiple myeloma. *New England Journal of Medicine*, 349(26):2483–2494, 2003.

[VJR+19] Lokeswari Venkataramana, Shomona Gracia Jacob, Rajavel Ramadoss, Dodda Saisuma, Dommaraju Haritha, and Kunthipuram Manoja. Improving classification accuracy of cancer types using parallel hybrid feature selection on microarray gene expression data. *Genes & genomics*, pages 1–13, 2019.

[VVDVDV+02] Laura J Van't Veer, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin Van Der Kooy, Matthew J Marton, Anke T Witteveen, et al. Gene expression profiling predicts clinical outcome of breast cancer. *nature*, 415(6871):530–536, 2002.

[WDH+18] Luyu Wang, Gavin Weiguang Ding, Ruitong Huang, Yanshuai Cao, and Yik Chau Lui. Adversarial robustness of pruned neural networks. 2018.

[Wes00] David West. Neural network credit scoring models. *Computers & Operations Research*, 27(11-12):1131–1152, 2000.

[WMF+19] Qing Wu, Zheping Ma, Jin Fan, Gang Xu, and Yuanfeng Shen. A feature selection method based on hybrid improved binary quantum particle swarm optimization. *IEEE Access*, 7:80588–80601, 2019.

[WWW+16] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[WWWY15] Ruobing Wu, Baoyuan Wang, Wenping Wang, and Yizhou Yu. Harvesting discriminative meta objects with deep cnn features for scene classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1287–1295, 2015.

[WYT+07] Xiangyang Wang, Jie Yang, Xiaolong Teng, Weijun Xia, and Richard Jensen. Feature selection based on rough sets and particle swarm optimization. *Pattern recognition letters*, 28(4):459–471, 2007.

[WZZ13] Bo Wu, Liangpei Zhang, and Yindi Zhao. Feature selection via cramer's v-test discretization for remote-sensing image classification. *IEEE transactions on geoscience and remote sensing*, 52(5):2593–2606, 2013.

[XH19] Ying Xiong and Fei Han. A hybrid gene selection method for microarray data based on geodesic distance and binary particle swarm optimization. In *IOP Conference Series: Materials Science and Engineering*, volume 490, page 042014. IOP Publishing, 2019.

[XHE+10] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[XHH+17] Gui-Song Xia, Jingwen Hu, Fan Hu, Baoguang Shi, Xiang Bai, Yanfei Zhong, Liangpei Zhang, and Xiaoqiang Lu. Aid: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, 2017.

[XZW+16] Lingxi Xie, Liang Zheng, Jingdong Wang, Alan L Yuille, and Qi Tian. Interactive: Inter-layer activeness propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2016.

[YCKY08] Cheng-San Yang, Li-Yeh Chuang, Chao-Hsuan Ke, and Cheng-Hong Yang. A hybrid feature selection method for microarray classification. *IAENG International Journal of Computer Science*, 35(3), 2008.

[YCLL06] Kun Yang, Zhipeng Cai, Jianzhong Li, and Guohui Lin. A stable gene selection in microarray data analysis. *BMC bioinformatics*, 7(1):228, 2006.

[YCS17] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017.

[YCY+10] Cheng-Huei Yang, Li-Yeh Chuang, C Hong Yang, et al. Ig-ga: a hybrid filter/wrapper method for feature selection of microarray data. *Journal of Medical and Biological Engineering*, 30(1):23–28, 2010.

[YGL+09] Hualong Yu, Guochang Gu, Haibo Liu, Jing Shen, and Jing Zhao. A modified ant colony optimization algorithm for tumor marker gene selection. *Genomics, proteomics & bioinformatics*, 7(4):200–208, 2009.

[YL03a] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[YL03b] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[YL04a] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

[YL04b] Lei Yu and Huan Liu. Redundancy based feature selection for microarray data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–742, 2004.

[YL18] Yunlong Yu and Fuxian Liu. A two-stream deep fusion framework for high-resolution aerial scene classification. *Computational intelligence and neuroscience*, 2018, 2018.

[YLC+18] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[YN08] Yi Yang and Shawn Newsam. Comparing sift descriptors and gabor texture features for classification of remote sensed im-

agery. In *2008 15th IEEE international conference on image processing*, pages 1852–1855. IEEE, 2008.

[YN10] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279. ACM, 2010.

[YWW+18] Shaokai Ye, Siyue Wang, Xiao Wang, Bo Yuan, Wujie Wen, and Xue Lin. Defending dnn adversarial attacks with pruning and logits augmentation. 2018.

[YYX+18] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[ZC19] Haitao Zeng and Gongwei Chen. Scene recognition with comprehensive regions graph modeling. In *International Conference on Image and Graphics*, pages 630–641. Springer, 2019.

[ZDZ14] Fan Zhang, Bo Du, and Liangpei Zhang. Saliency-guided unsupervised feature learning for scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):2175–2184, 2014.

[ZDZ15] Fan Zhang, Bo Du, and Liangpei Zhang. Scene classification via a gradient boosting random convolutional network framework. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3):1793–1802, 2015.

[ZFZ16] Yanfei Zhong, Feng Fei, and Liangpei Zhang. Large patch convolutional neural networks for the scene classification of high spatial resolution imagery. *Journal of Applied Remote Sensing*, 10(2):025006, 2016.

[ZL09] Zheng Zhao and Huan Liu. Searching for interacting features in subset selection. *Intelligent Data Analysis*, 13(2):207–228, 2009.

[ZL18] Zhengyu Zhao and Martha Larson. From volcano to toyshop: Adaptive discriminative region discovery for scene recognition. *arXiv preprint arXiv:1807.08624*, 2018.

[ZLCD16] Jinyi Zou, Wei Li, Chen Chen, and Qian Du. Scene classification using local and global features with collaborative representation fusion. *Information Sciences*, 348:209–226, 2016.

[ZLX+14] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[ZNZ+19] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2019.

[ZRZ+18] Junhua Zou, Ting Rui, You Zhou, Chengsong Yang, and Sai Zhang. Convolutional neural network simplification via feature map pruning. *Computers & Electrical Engineering*, 70:950–958, 2018.

[ZS15] Masoumeh Zareapoor and KR Seeja. Feature extraction or feature selection for text classification: A case study on phishing email detection. *International Journal of Information Engineering and Electronic Business*, 7(2):60, 2015.

[ZTZ19] Wei Zhang, Ping Tang, and Lijun Zhao. Remote sensing image scene classification using cnn-capsnet. *Remote Sensing*, 11(5):494, 2019.

[ZWLY11] Zheng Zhao, Lei Wang, Huan Liu, and Jieping Ye. On similarity preserving feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):619–632, 2011.

[ZWS+14] Zhen Zuo, Gang Wang, Bing Shuai, Lifan Zhao, Qingxiong Yang, and Xudong Jiang. Learning discriminative and shareable features for scene classification. In *European Conference on Computer Vision*, pages 552–568. Springer, 2014.

[ZZD16] Liangpei Zhang, Lefei Zhang, and Bo Du. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):22–40, 2016.

[ZZLS18] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[ZZZ16a] Bei Zhao, Yanfei Zhong, and Liangpei Zhang. A spectral–structural bag-of-features scene classifier for very high spatial resolution remote sensing imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:73–85, 2016.

[ZZZ+16b] Qiqi Zhu, Yanfei Zhong, Bei Zhao, Gui-Song Xia, and Liangpei Zhang. Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery. *IEEE Geoscience and Remote Sensing Letters*, 13(6):747–751, 2016.

Appendix A

IRRD-GA Full Related Work Results

Table A.1: Comparison of State-of-the-art results using 3-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Breast	[NM09b]	4	79	2016
Breast	[BCSMAB15]	18	100	2016
CNS	[SPBCAB16]	47	60.00	2016
CNS	[SPBCAB16]	60	65.00	2016
CNS	[SPBCAB16]	25	65.00	2016
CNS	[SPBCAB16]	3	65.00	2016
CNS	[SPBCAB16]	25	70.00	2016
CNS	[SPBCAB16]	5	75.00	2016
Colon	[TNRM15]	20	80	2016
Colon	[SPBCAB16]	3	80.00	2016
Colon	[SPBCAB16]	5	85.00	2016
Colon	[SPBCAB16]	19	85.00	2016
Colon	[SPBCAB16]	16	85.00	2016
Colon	[SPBCAB16]	25	85.00	2016
Colon	[SPBCAB16]	25	85.00	2016
Leukemia2	[SPBCAB16]	1	79.41	2016
Leukemia2	[SPBCAB16]	25	82.35	2016
Leukemia2	[SPBCAB16]	36	88.24	2016
Leukemia2	[SPBCAB16]	25	88.24	2016
Leukemia2	[SPBCAB16]	5	91.18	2016
Leukemia2	[TNRM15]	20	92.31	2016

Table A.2: Comparison of State-of-the-art results using 3-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Leukemia2	[KSM15]	3	95.89	2015
Leukemia6	[LZO04]	150	67.00	2011
Leukemia6	[KSM15]	4	92.54	2015
Leukemia6	[SIM11]	4	93.00	2011
Leukemia6	[LZO04]	150	100	2011
Leukemia6	[SIM11]	4	100	2011
Lung2	[SPBCAB16]	1	81.88	2016
Lung2	[SPBCAB16]	25	97.99	2016
Lung2	[SPBCAB16]	40	98.66	2016
Lung2	[SPBCAB16]	25	99.33	2016
Lung2	[SPBCAB16]	5	99.33	2016
Lung4	[TNRM15]	20	85.72	2016
Lymphoma1	[SPBCAB16]	36	86.67	2016
Lymphoma1	[SPBCAB16]	2	86.67	2016
Lymphoma1	[SPBCAB16]	47	93.33	2016
Lymphoma1	[SPBCAB16]	25	93.33	2016
Lymphoma1	[SPBCAB16]	5	93.33	2016
Ovarian	[SPBCAB16]	25	98.81	2016
Ovarian	[SPBCAB16]	37	100	2016
Ovarian	[SPBCAB16]	25	100	2016
Ovarian	[SPBCAB16]	5	100	2016
Ovarian	[SPBCAB16]	3	100	2016
Ovarian	[SPBCAB16]	2	100	2016
Prostate1	[TNRM15]	20	73.15	2016
Prostate3	[SPBCAB16]	4	26.53	2016
Prostate3	[SPBCAB16]	73	70.59	2016
Prostate3	[SPBCAB16]	5	73.53	2016
Prostate3	[SPBCAB16]	25	94.12	2016
Prostate3	[SIM11]	4	97.00	2011
Prostate3	[SPBCAB16]	89	97.06	2016
Prostate3	[SPBCAB16]	25	97.06	2016
Prostate3	[SIM11]	4	100	2011
SRBCT	[LZO04]	150	68.00	2011
SRBCT	[TNRM15]	20	84.14	2016
SRBCT	[SIM11]	4	90.00	2011
SRBCT	[LZO04]	150	95.00	2011
SRBCT	[SIM11]	4	95.00	2011
SRBCT	[KSM15]	6	98.01	2015
SRBCT	[SIM11]	4	100	2011

Table A.3: Comparison of State-of-the-art results using 5-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Brain	[BCSMAB+14c]	10	91	2014
Bone	[GZL+16]	7	85	2016
Breast	[GBS+19]	50	88.82	2018
CNS	[BCSMAB+14c]	10	72	2014
CNS	[GZL+16]	4	81.5	2016
CNS	[XH19]	9	92.34	2019
Colon	[BCSMAB+14c]	50	85	2014
Colon	[GZL+16]	6	89	2016
Colon	[LCT02]	15	93.6	2008
Colon	[MNSMN17]	9	99.81	2017
Leukemia1	[BCSMAB+14c]	50	92	2014
Leukemia2	[YCLL06]	93	97.9	2006
Leukemia2	[GZL+16]	2	99.5	2016
Leukemia2	[LCT02]	4	100	2008
Leukemia2	[MNSMN17]	2	100	2017
Leukemia5	[GBS+19]	4	98.67	2019
Leukemia6	[MNSMN17]	3	93.96	2017
Leukemia6	[YCLL06]	93	96	2006
Leukemia6	[GBS+19]	4	96	2019
Leukemia6	[SIM11]	4	100	2011
Lung3	[BCSMAB+14c]	50	72	2014
Lung4	[XH19]	12	97.45	2019
Lymphoma1	[GZL+16]	4	99.5	2016
Lymphoma1	[BCSMAB+14c]	10	98	2014
Lymphoma2	[GBS+19]	3	98.75	2019
Lymphoma3	[XH19]	9	92.23	2019
Ovarian	[BCSMAB+14c]	31	100	2014
Prostate2	[GZL+16]	3	93	2016
Prostate4	[GBS+19]	3	98.10	2019
SRBCT	[YCLL06]	95	99.2	2006
SRBCT	[MNSMN17]	6	99.34	2017
SRBCT	[SIM11]	4	100	2011
SRBCT	[GBS+19]	5	100	2019

Table A.4: Comparison of State-of-the-art results using 10-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Bone	[WMF+19]	38.8	83.71	2019
Brain	[BSZ17]	50	84.7	2017
Breast	[YL04b]	67	79.38	2004
Breast	[TJGNA08]	4	86.35	2008
Breast	[GB13]	6	86.90	2018
Breast	[DPHC18]	4.2	92.89	2018
CNS	[BSZ17]	50	56.4	2017
CNS	[WMF+19]	36.28	73.64	2019
CNS	[SANA12]	32	76.6	2012
CNS	[SANA12]	31	78	2012
CNS	[Das17]	5	83	2017
CNS	[SANA12]	21.8	90.2	2012
CNS	[BHDH+11]	3	99.3	2011
Colon	[MZO17]	82.1	85.48	2017
Colon	[BSZ17]	50	84.7	2017
Colon	[WMF+19]	24.8	88.33	2019
Colon	[BCSMAB15]	10	90	2015
Colon	[YL04b]	4	93.55	2004
Colon	[ALA16]	3	93.80	2016
Colon	[DR03]	7	97	2003
Colon	[KRMV15]	5	98.40	2015
Colon	[DPHC18]	2	99.16	2018
Colon	[KAAAJ11]	19	100	2011
Colon	[MM16]	10	100	2016
Colon	[Das17]	5	100	2017
Colon	[TJGNA08]	2	100	2008
Leukemia1	[WMF+19]	12.92	100	2019
Leukemia2	[YL04b]	4	87.55	2004
Leukemia2	[BCSMAB15]	13	91.18	2015
Leukemia2	[ALA16]	1	94.10	2016
Leukemia2	[TJGNA08]	3	97.38	2008
Leukemia2	[SMM16]	5.25	97.60	2016
Leukemia2	[WMF+19]	21.32	97.81	2019
Leukemia2	[KRMV15]	3	99.01	2015
Leukemia2	[DPHC18]	2.4	99.50	2018
Leukemia2	[DPHC18]	2.4	99.50	2018
Leukemia2	[DR03]	4	100	2003

Table A.5: Comparison of State-of-the-art results using 10-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Leukemia3	[KAAAJ11]	10	100	2011
Leukemia4	[SMM16]	13.08	86.38	2016
Leukemia6	[SMM16]	14.9	99.30	2016
Lung1	[MZO17]	206	100	2017
Lung1	[Das17]	5	100	2017
Lung2	[ALA16]	3	98	2016
Lung2	[YL04b]	6	98.34	2004
Lung2	[BCSMAB15]	17	98.66	2015
Lung2	[TJGNA08]	4	99	2008
Lung2	[KRMV15]	4	99.40	2015
Lung2	[DPHC18]	4.1	100	2018
Lung3	[BSZ17]	50	73.5	2017
Lung4	[KAAAJ11]	40	99.5	2011
Lung4	[MM16]	10	100	2016
Lymphoma1	[SANA12]	29	91.6	2012
Lymphoma1	[BCSMAB15]	11	93.33	2015
Lymphoma1	[SANA12]	31.2	98.9	2012
Lymphoma1	[BHDH$^+$11]	3	99.5	2011
Lymphoma1	[KAAAJ11]	23	100	2011
Lymphoma1	[Das17]	6	100	2017
Lymphoma1	[SANA12]	2.8	100	2012
Lymphoma1	[ALA16]	2	100	2016
Lymphoma2	[WMF$^+$19]	20.56	94.50	2019
Lymphoma2	[GBGK12]	30	95	2012
Lymphoma3	[CKY16]	107	97.14	2016
Lymphoma3	[CKY16]	33	98.57	2016
Lymphoma3	[MM16]	10	100	2016

Table A.6: Comparison of State-of-the-art results using 10-fold cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Ovarian	[TJGNA08]	4	99.44	2008
Ovarian	[BCSMAB15]	16	100	2015
Ovarian	[ALA16]	2	100	2016
Prostate1	[BSZ17]	50	95.2	2017
Prostate1	[MM16]	20	96.78	2016
Prostate1	[KAAAJ11]	45	99.6	2011
Prostate2	[SANA12]	25	85.2	2012
Prostate2	[SANA12]	30	86.4	2012
Prostate2	[WMF+19]	24.84	91.60	2019
Prostate2	[GBGK12]	30	94	2012
Prostate2	[BHDH+11]	3	99.5	2011
Prostate2	[Das17]	3	100	2017
Prostate2	[SANA12]	2.2	100	2012
Prostate2	[DPHC18]	2	100	2018
Prostate3	[MZO17]	8	71.43	2017
Prostate3	[BSZ17]	50	80.7	2017
Prostate3	[BCSMAB15]	113	85.29	2015
Prostate3	[ALA16]	3	97.1	2016
Prostate3	[TJGNA08]	4	98.66	2008
Prostate3	[SMM16]	14.5	99.85	2016
Prostate4	[MM13]	-	96.08	2013
SRBCT	[BSZ17]	50	100	2017
SRBCT	[MM16]	5	100	2016

Table A.7: Comparison of State-of-the-art results using leave-one-out cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Breast	[TES17]	5127	100	2017
CNS	[SAEF17]	38	86.67	2017
Colon	[SAEF17]	60	85.48	2017
Colon	[SSKY07]	20	85.48	2007
Colon	[AHR12]	10	90.32	2012
Colon	[TES17]	240	91.97	2017
Colon	[LH08]	6	95.16	2008
Colon	[ABA15a]	15	96.77	2015
Colon	[ABA15b]	10	98.38	2015
Leukemia2	[LJL+14]	50	80.56	2014
Leukemia2	[SSKY07]	23	94.44	2007
Leukemia2	[SAEF17]	3	97.06	2017
Leukemia2	[DB17]	12	98.10	2017
Leukemia2	[YCLL06]	4	98.6	2006
Leukemia2	[AHR12]	18	100	2012
Leukemia2	[ABA15a]	14	100	2015
Leukemia2	[LH08]	4	100	2008
Leukemia2	[ABA15b]	4	100	2015
Leukemia3	[YCKY08]	41	98.57	2008
Leukemia3	[ABA15a]	20	100	2015
Leukemia3	[HLH07]	9	100	2007
Leukemia3	[ABA15b]	8	100	2015
Leukemia5	[TES17]	224	98.61	2017
Leukemia5	[GBS+19]	4	98.61	2019
Leukemia6	[LJL+14]	15	79.17	2014
Leukemia6	[GBS+19]	4	95.83	2019
Leukemia6	[YCLL06]	23	97.2	2006
Leukemia6	[MODY11]	4	100	2011
Lung1	[YCY+10]	9561	90.15	2010
Lung1	[YCY+10]	1845	94.09	2010
Lung1	[YCY+10]	2101	95.57	2010
Lung1	[CYWY11]	195.2	98.42	2011
Lung1	[HC07]	7	95.75	2007
Lung1	[CYWY11]	1897	96.55	2011
Lung1	[HLH07]	10	97.1	2007
Lung1	[SAEF17]	9	100	2017
Lung1	[ABA15a]	8	100	2015
Lung1	[ABA15b]	4	100	2015
Lung2	[LH08]	6	98.34	2010
Lung2	[LJL+14]	30	99.45	2014

Table A.8: Comparison of State-of-the-art results using leave-one-out cross-validation.

Dataset	Reference	#Features	Accuracy	Year
Lymphoma1	[DB17]	21	99.70	2017
Lymphoma2	[LJL+14]	10	81.82	2014
Lymphoma2	[GBGK12]	30	94	2012
Lymphoma2	[SAEF17]	110	94.80	2017
Lymphoma2	[GBS+19]	3	98.70	2019
Lymphoma3	[YCY+10]	1244	93.51	2010
Lymphoma3	[YCY+10]	882	93.51	2010
Lymphoma3	[YCY+10]	107	100	2010
Lymphoma3	[CKY16]	33	100	2016
Lymphoma3	[CYWY11]	17.1	100	2011
Lymphoma3	[LH08]	6	100	2008
Prostate1	[YCY+10]	2016	89.22	2010
Prostate1	[YCY+10]	3153	91.18	2010
Prostate1	[CYWY11]	1294	92.16	2011
Prostate1	[YCY+10]	343	96.08	2010
Prostate1	[CYWY11]	24.7	99.22	2011
Prostate2	[GBGK12]	30	94	2012
Prostate2	[LH08]	6	98.04	2010
Prostate3	[LJL+14]	25	73.53	2014
Prostate3	[DB17]	25	96.80	2017
Prostate3	[SAEF17]	26	100	2017
Prostate4	[GBS+19]	3	99.02	2019
SRBCT	[YCY+10]	536	96.39	2010
SRBCT	[DB17]	21	98.60	2017
SRBCT	[HC07]	6	98.75	2007
SRBCT	[GBS+19]	5	98.79	2019
SRBCT	[YCY+10]	669	98.80	2010
SRBCT	[LJL+14]	15	98.80	2014
SRBCT	[CYWY11]	431	100	2011
SRBCT	[YCLL06]	78	100	2006
SRBCT	[YCY+10]	56	100	2010
SRBCT	[CYWY11]	29	100	2011
SRBCT	[HLH07]	11	100	2007
SRBCT	[ABA15a]	10	100	2015
SRBCT	[ABA15b]	6	100	2015
SRBCT	[LL11]	6	100	2011

Appendix B

IRRD-GA Other Experiments Results

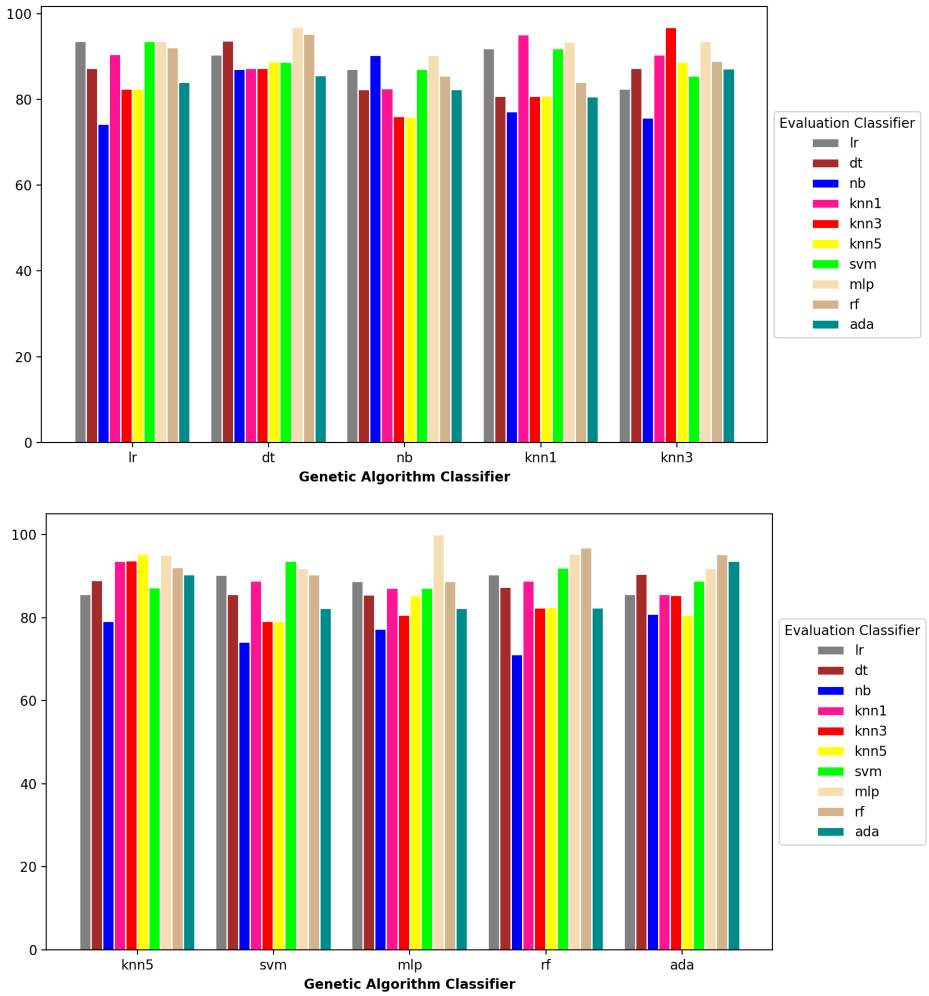

Figure B.1: Bar charts showing the results of using Cramer's ϕ statistic and a dominance ratio of 0.5 on the Colon dataset.

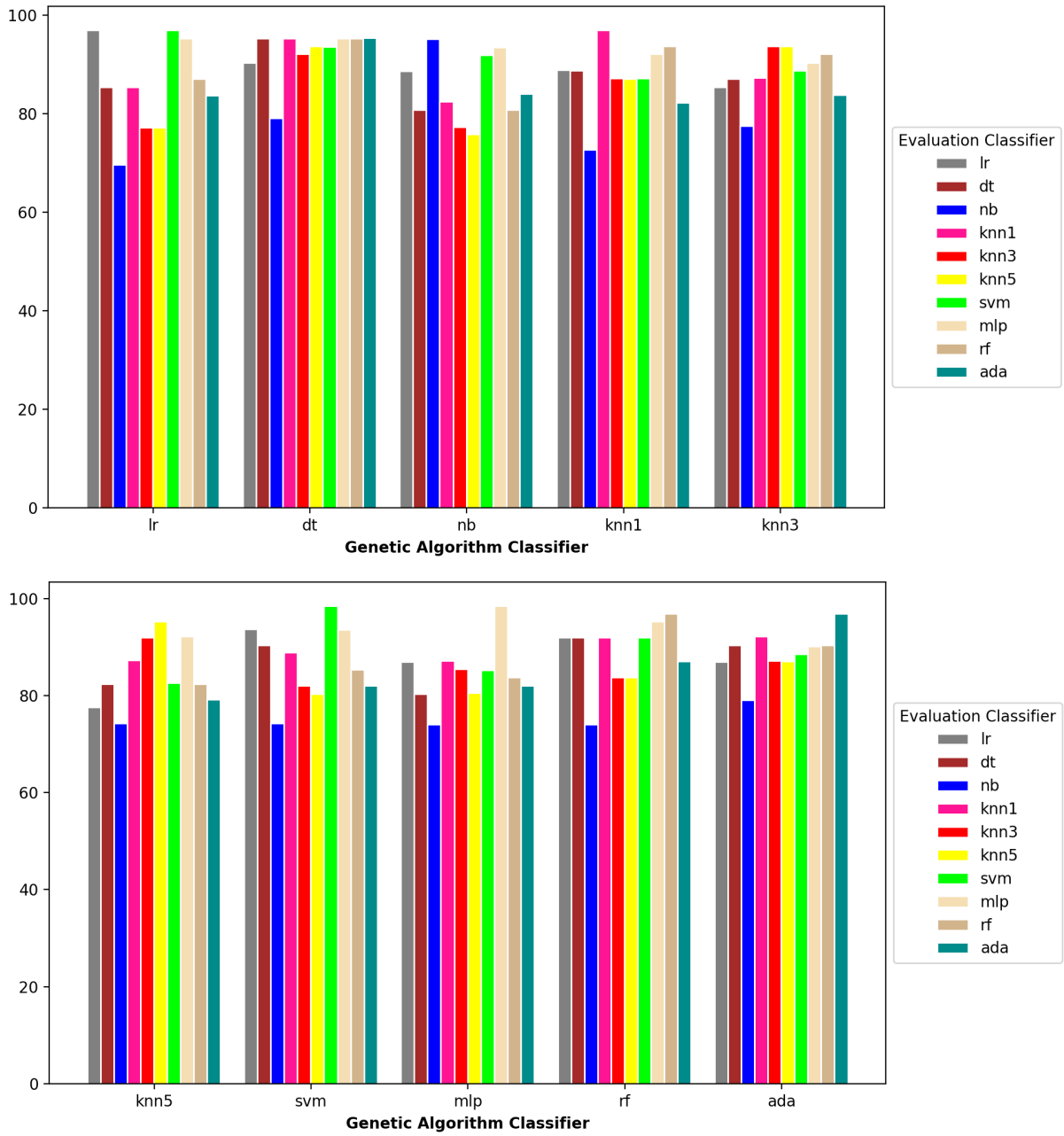

Figure B.2: Bar charts showing the results of using Cramer's ϕ statistic and a dominance ratio of 1.0 on the Colon dataset.

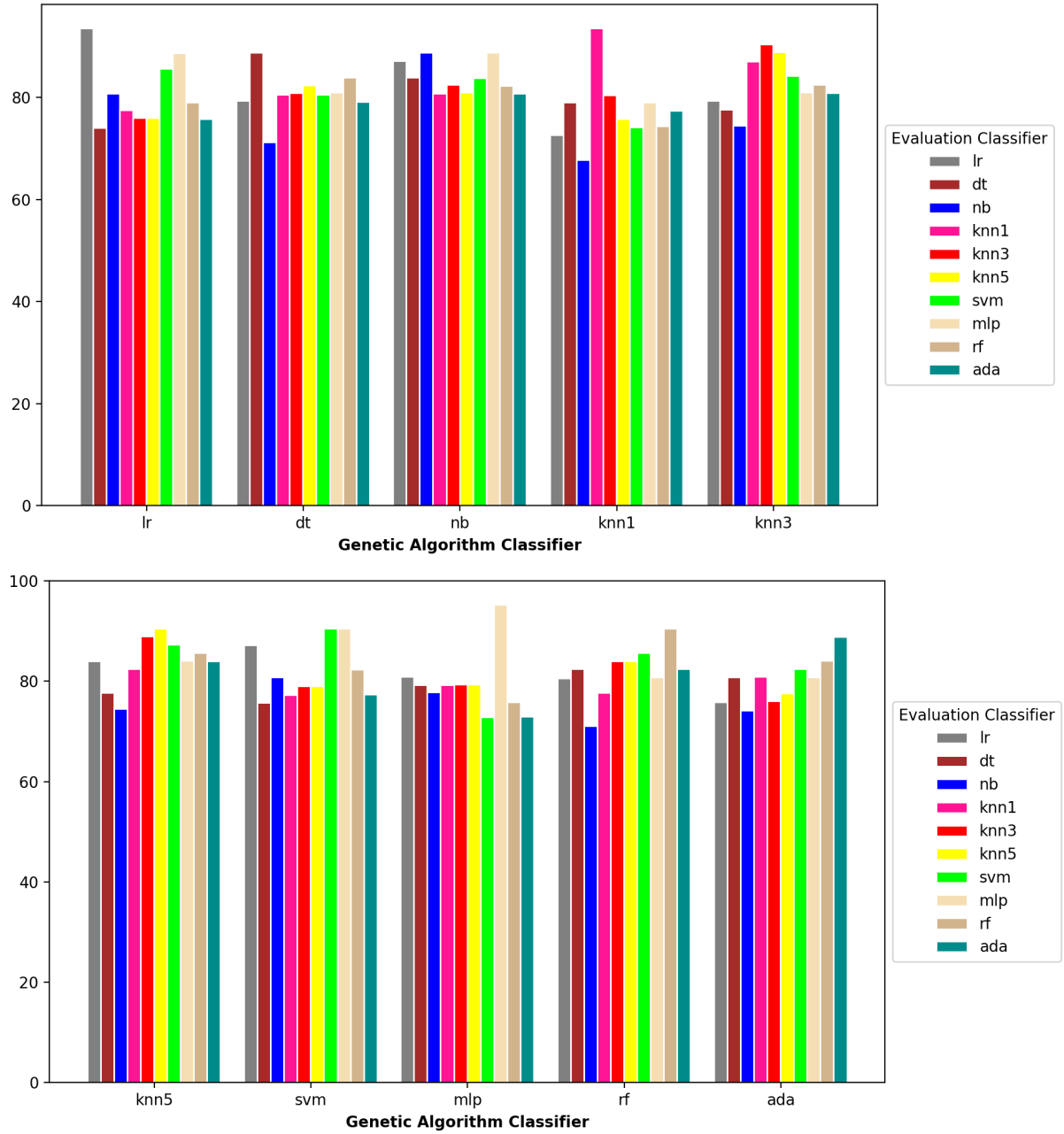

Figure B.3: Bar charts showing the results of using mutual information statistic and a dominance ratio of 0.5 on the Colon dataset.

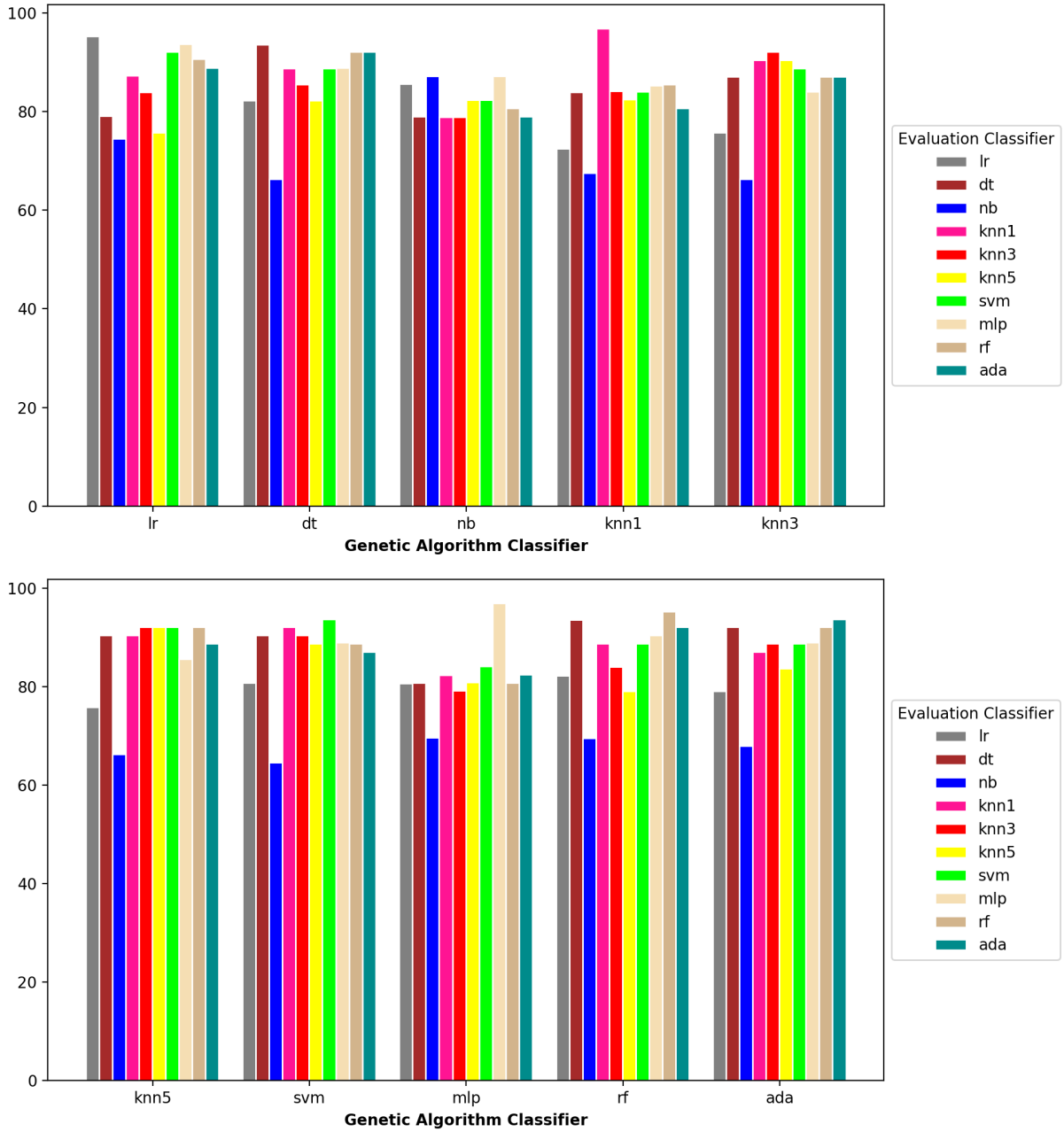

Figure B.4: Bar charts showing the results of using mutual information statistic and a dominance ratio of 1.0 on the Colon dataset.

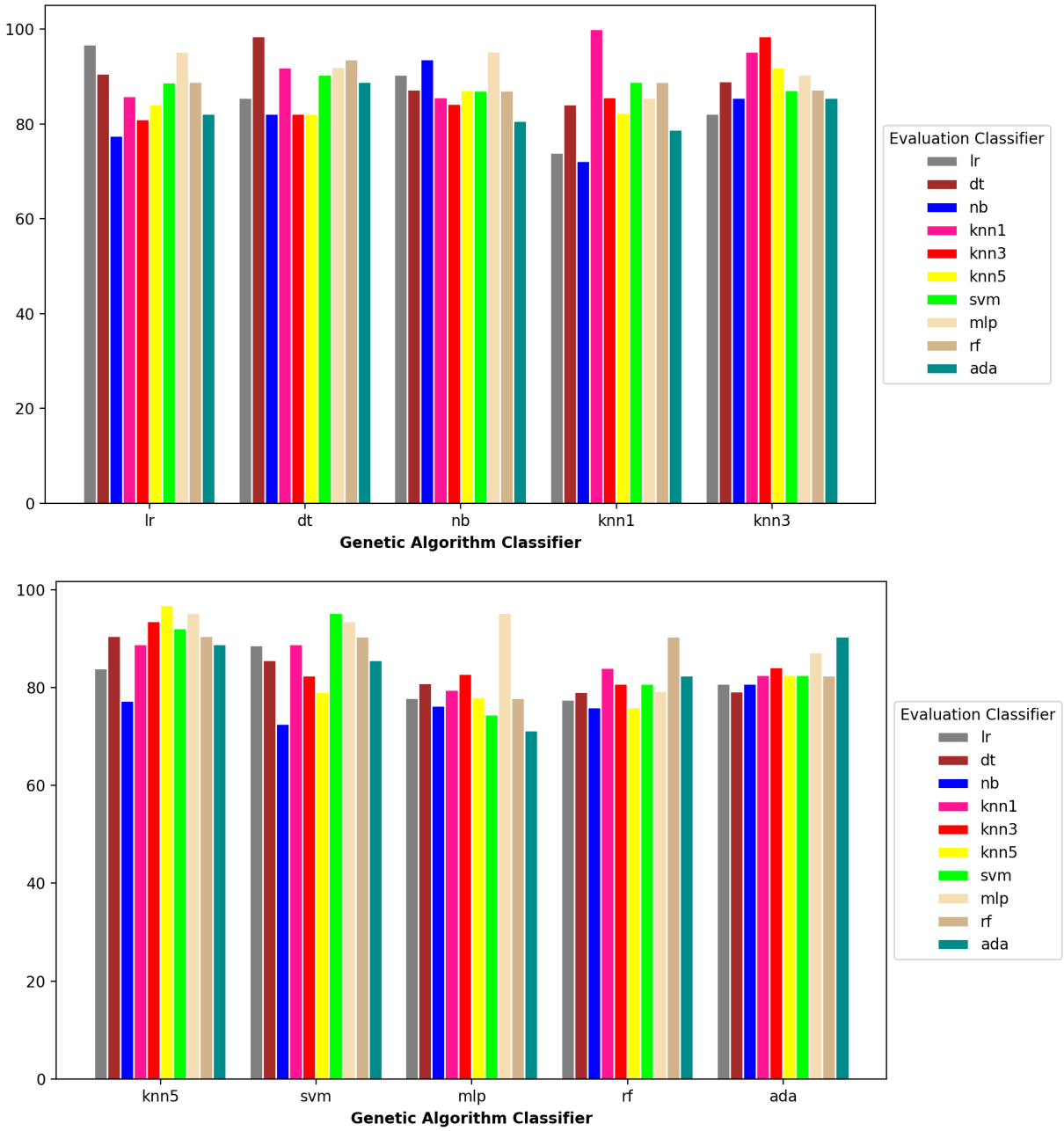

Figure B.5: Bar charts showing the results of using both statistics and a dominance ratio of 0.5 on the Colon dataset.

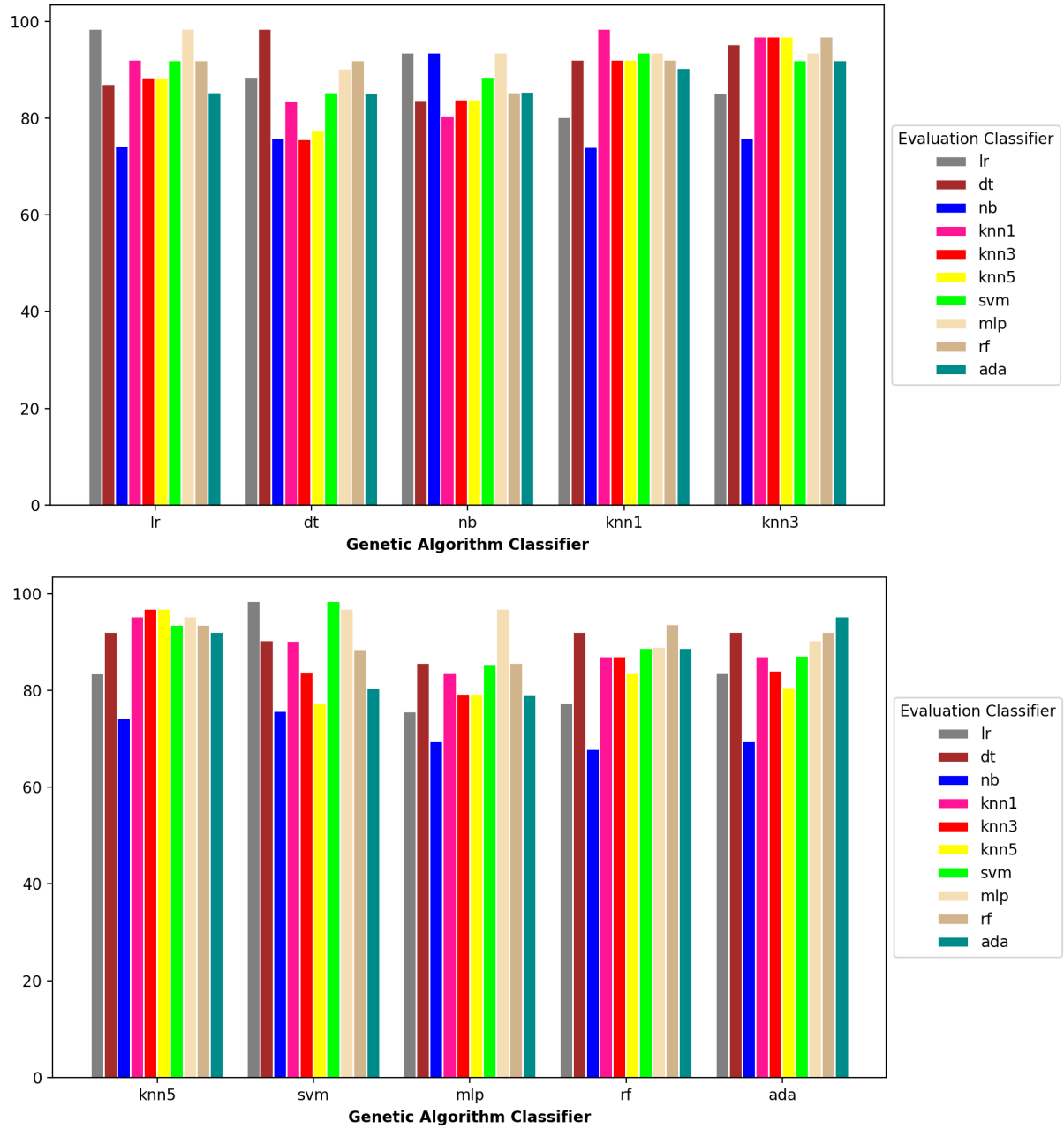

Figure B.6: Bar charts showing the results of using both statistics and a dominance ratio of 1.0 on the Colon dataset.

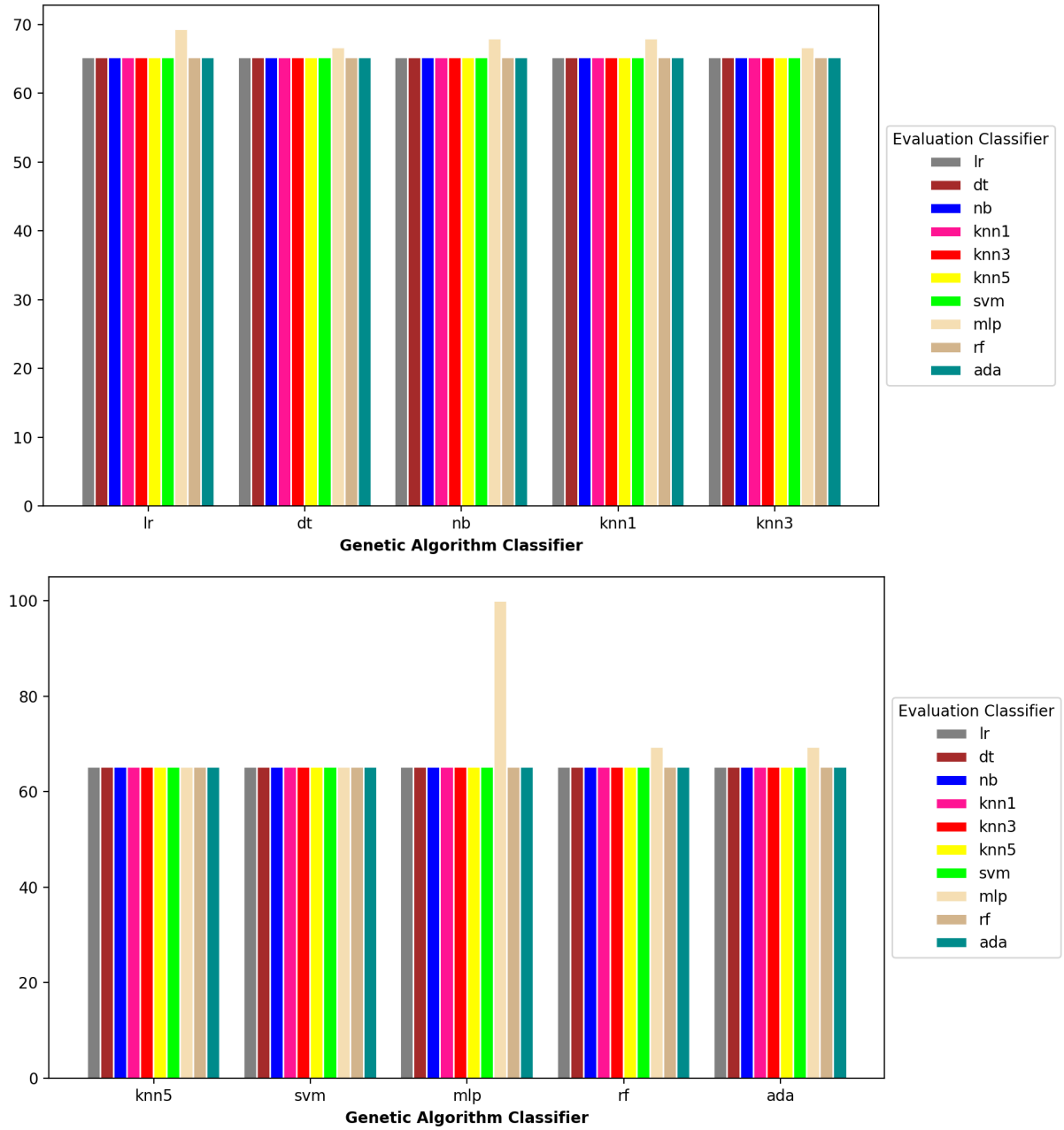

Figure B.7: Bar charts showing the results of using Cramer's ϕ statistic and a dominance ratio of 0.5 on the Leukemia2 dataset.

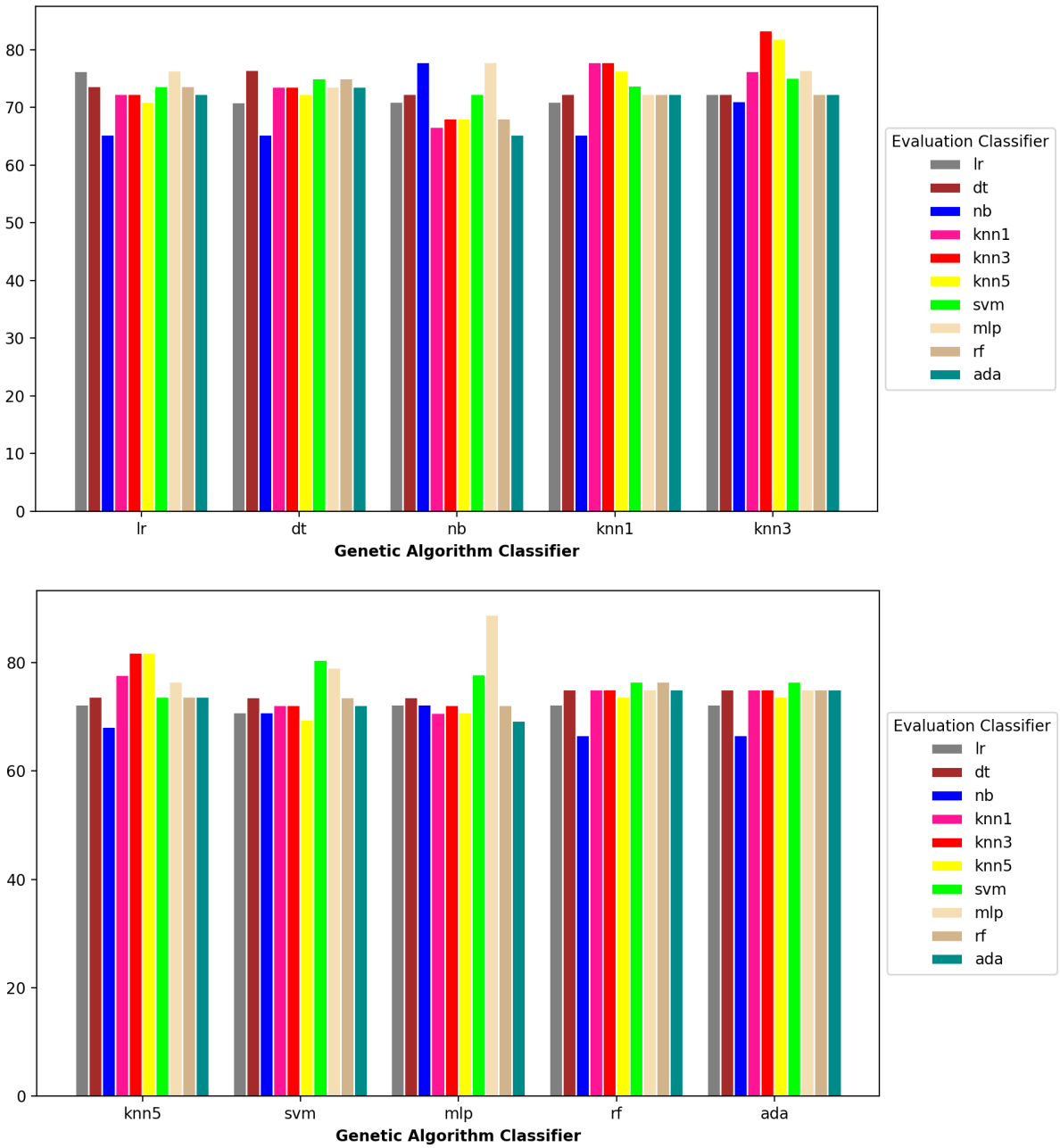

Figure B.8: Bar charts showing the results of using Cramer's ϕ statistic and a dominance ratio of 1.0 on the Leukemia2 dataset.

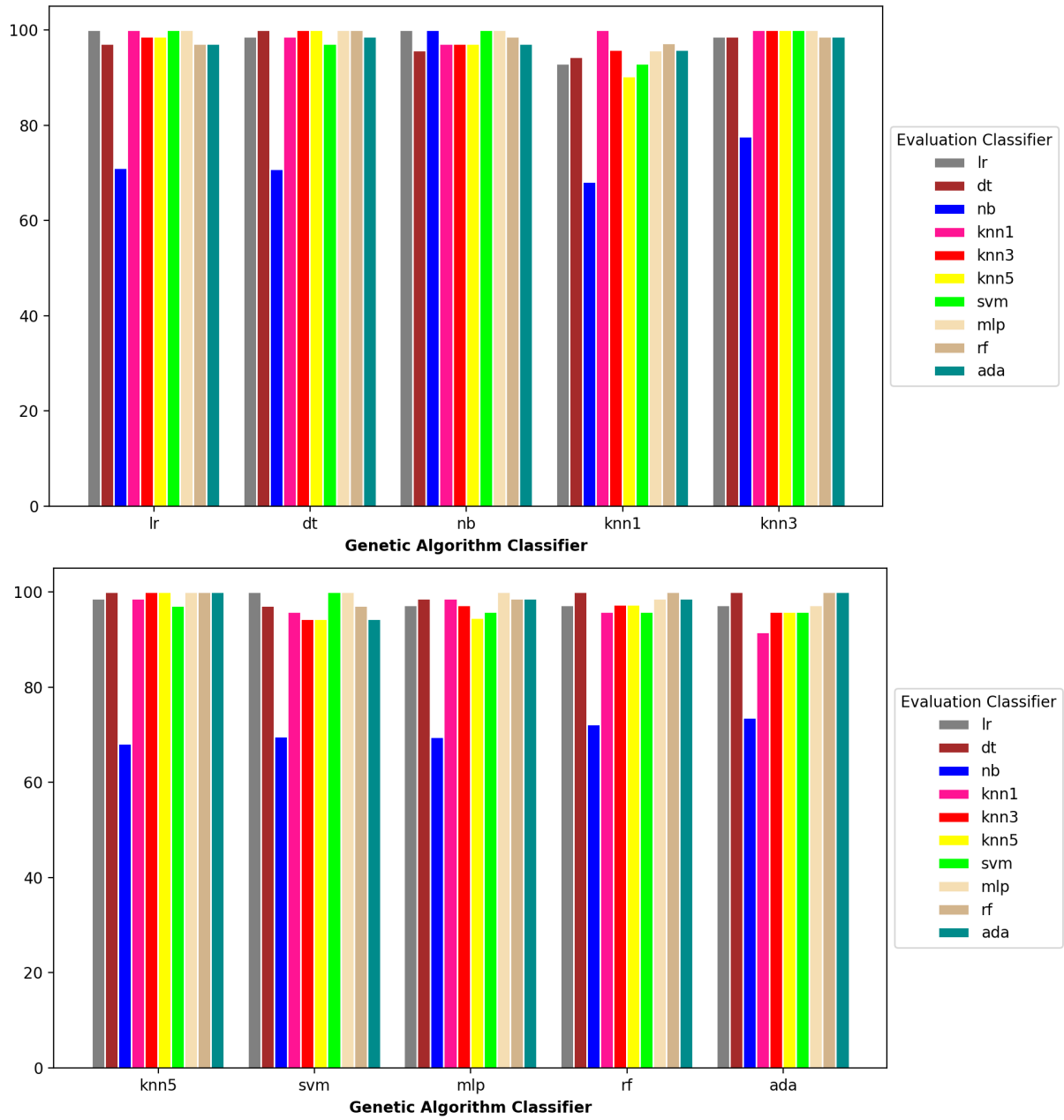

Figure B.9: Bar charts showing the results of using mutual information statistic and a dominance ratio of 0.5 on the Leukemia2 dataset.

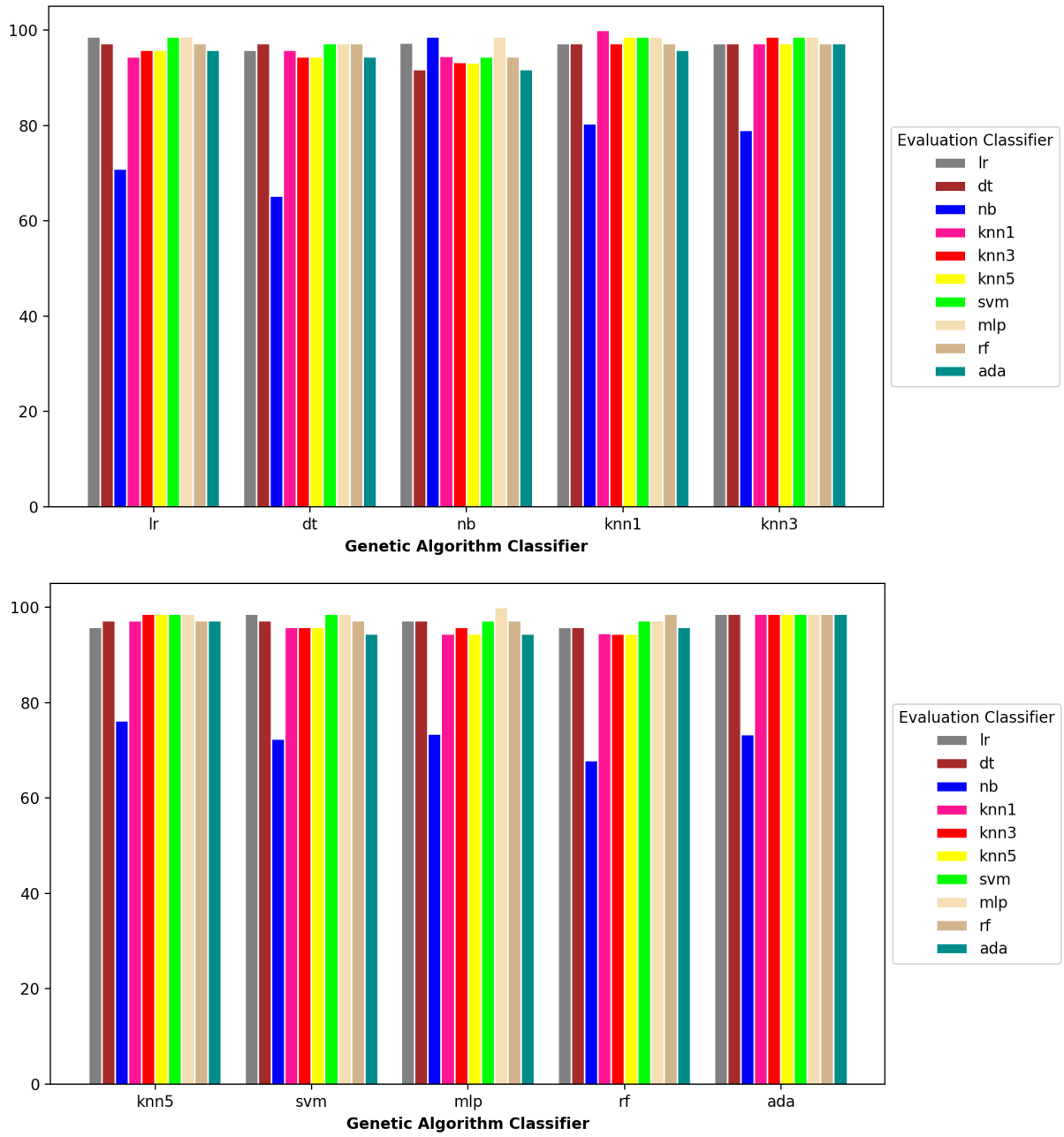

Figure B.10: Bar charts showing the results of using mutual information statistic and a dominance ratio of 1.0 on the Leukemia2 dataset.

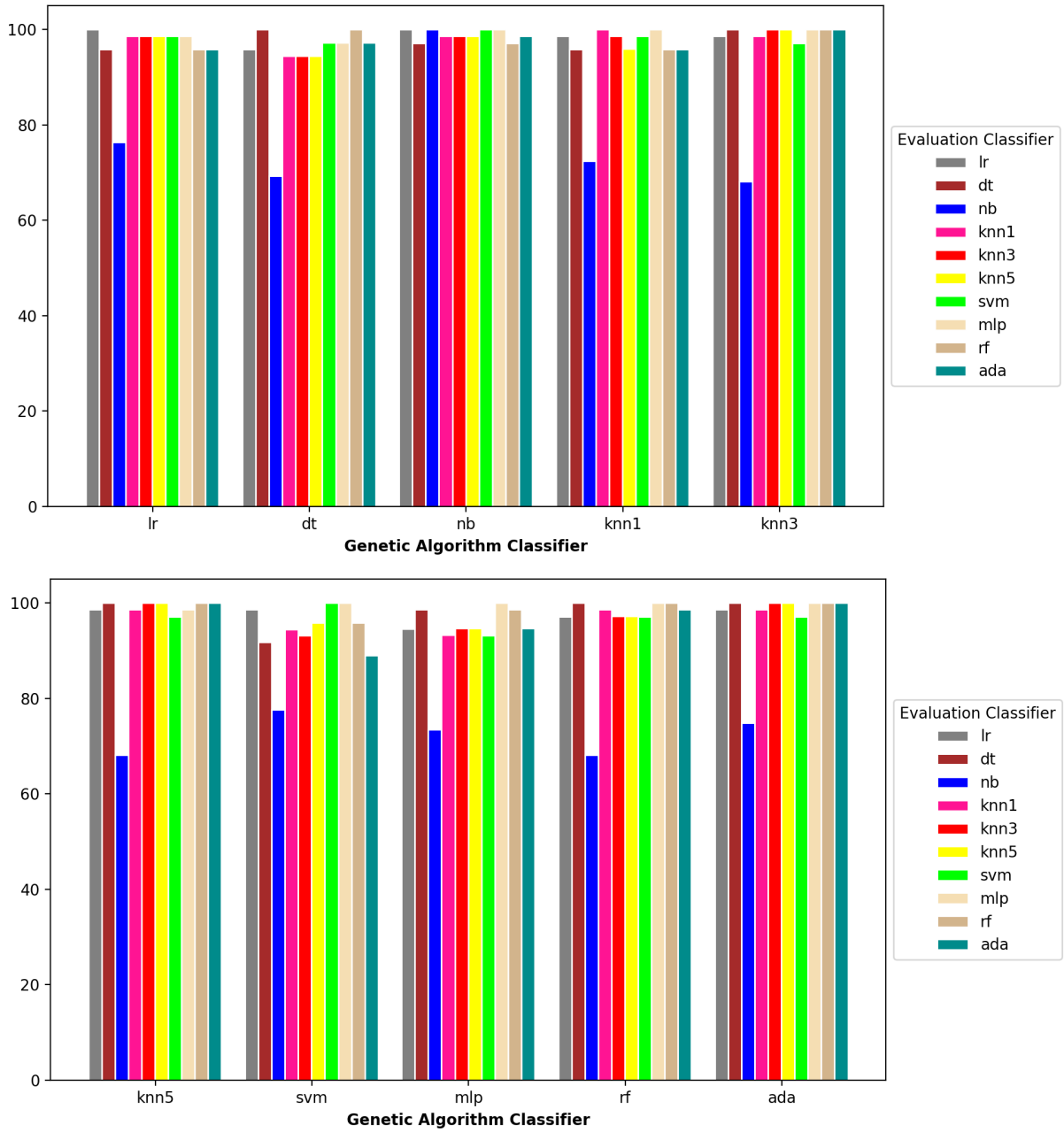

Figure B.11: Bar charts showing the results of using both statistics and a dominance ratio of 0.5 on the Leukemia2 dataset.

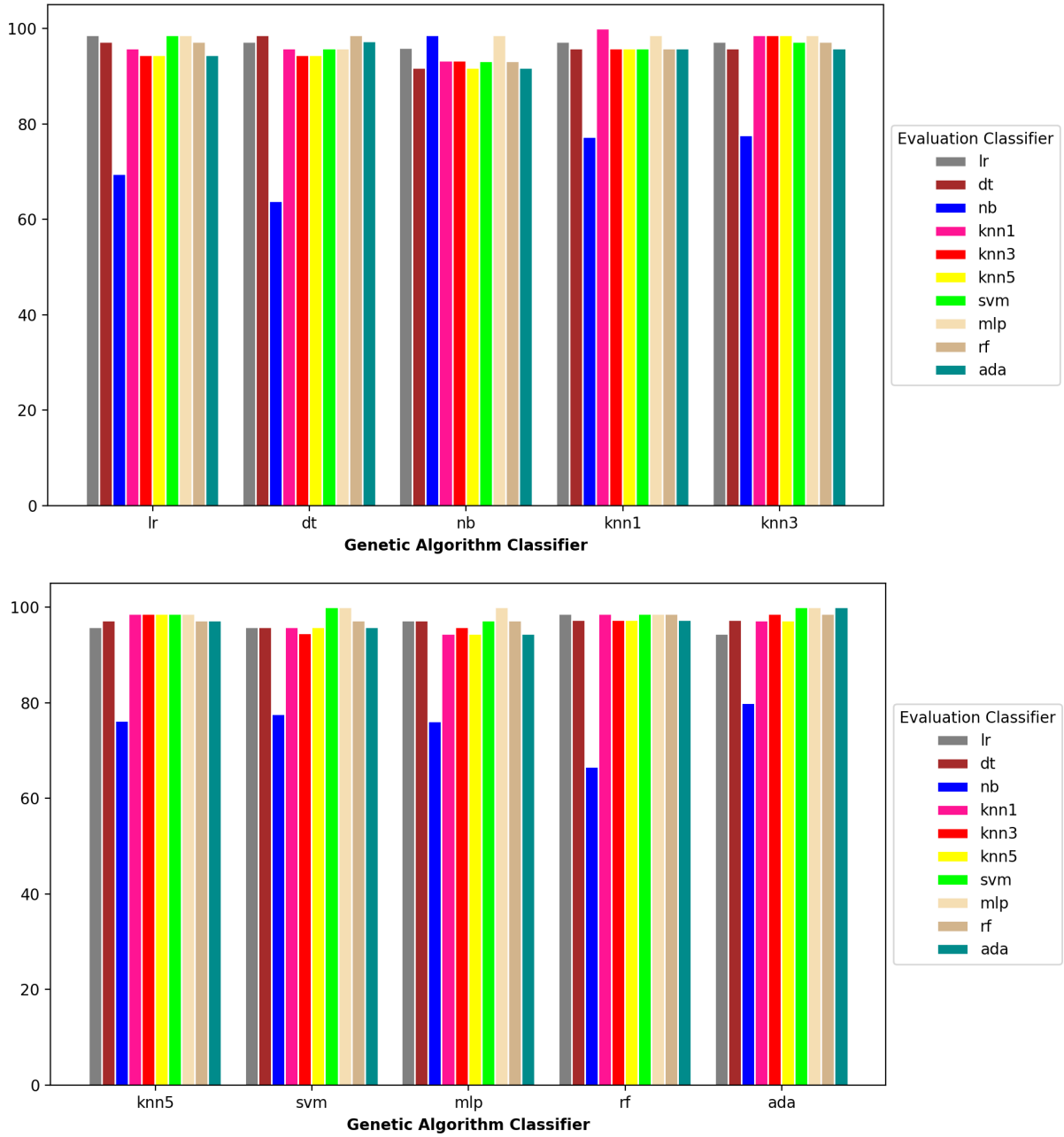

Figure B.12: Bar charts showing the results of using both statistics and a dominance ratio of 1.0 on the Leukemia2 dataset.

Appendix C

Unsupervised PulseNet Confusion Matrices Results

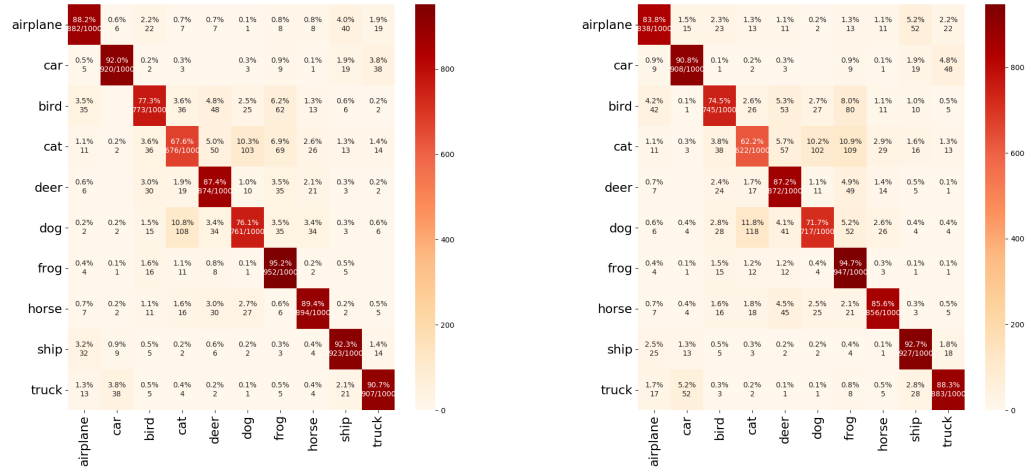

Figure C.1: Comparison of the confusion matrices between the original and pruned versions of CifarNet model on the CIFAR10 dataset.

C. Unsupervised PulseNet Confusion Matrices Results

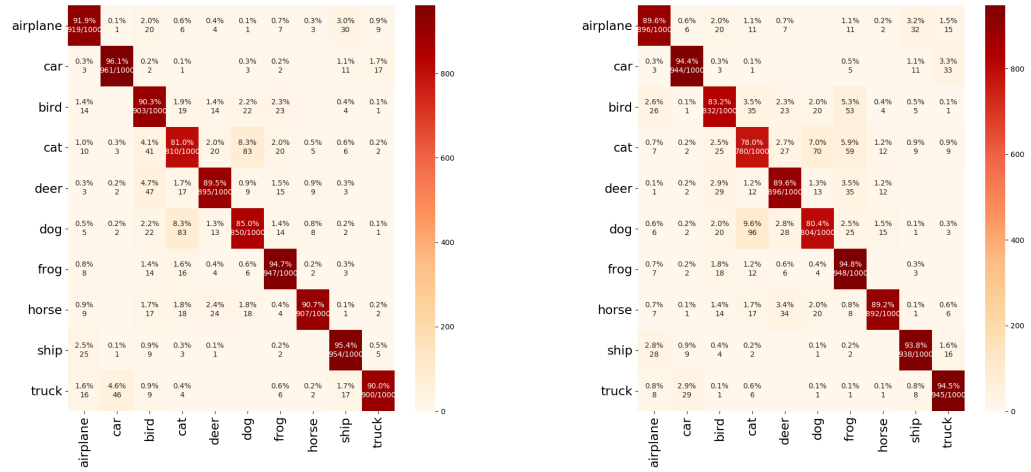

Figure C.2: Comparison of the confusion matrices between the original and pruned versions of AlexNet model on the CIFAR10 dataset.

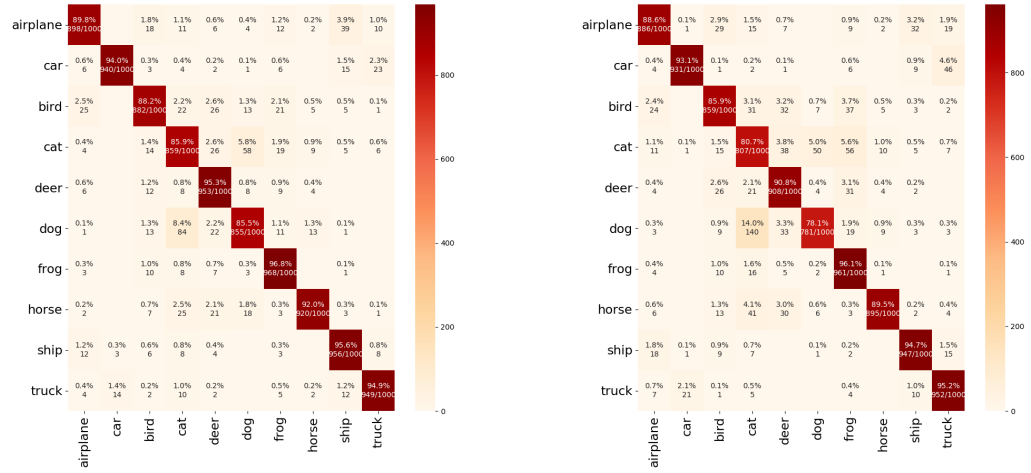

Figure C.3: Comparison of the confusion matrices between the original and pruned versions of VGG16 model on the CIFAR10 dataset.

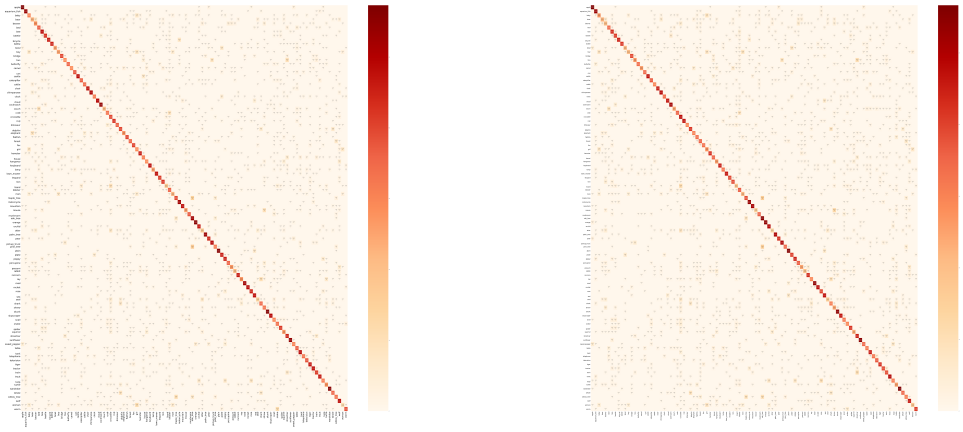

Figure C.4: Comparison of the confusion matrices between the original and pruned versions of CifarNet model on the CIFAR100 dataset.

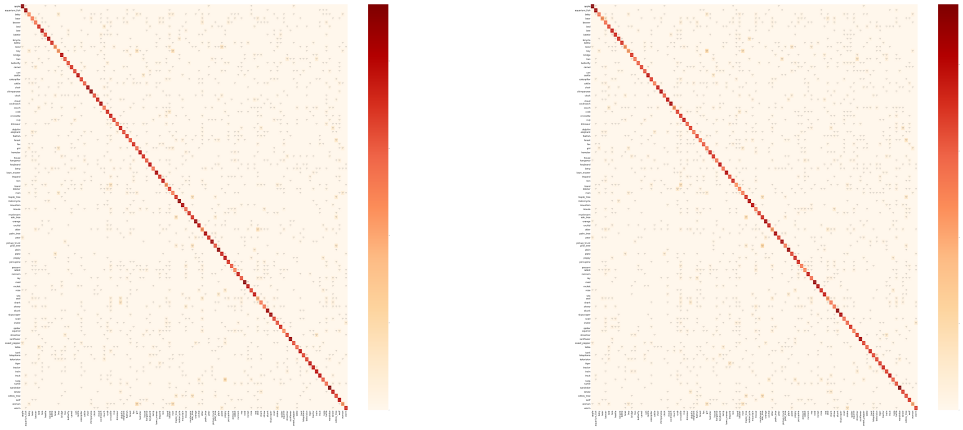

Figure C.5: Comparison of the confusion matrices between the original and pruned versions of AlexNet model on the CIFAR100 dataset.

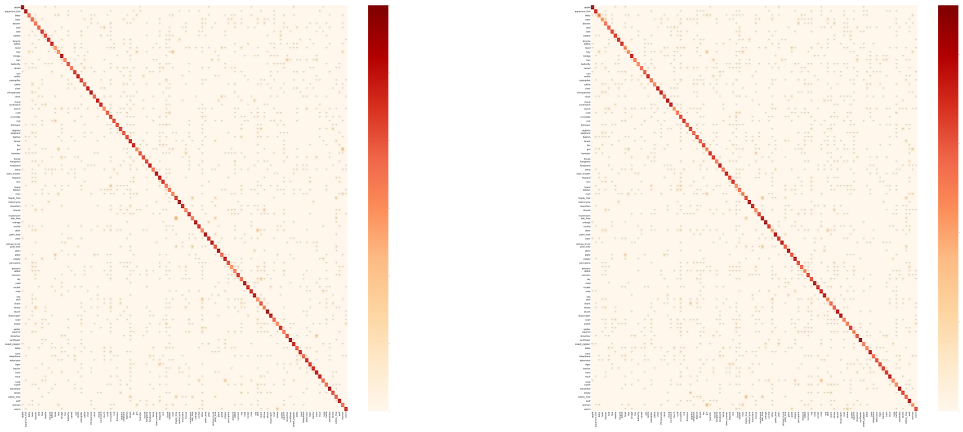

Figure C.6: Comparison of the confusion matrices between the original and pruned versions of VGG16 model on the CIFAR100 dataset.

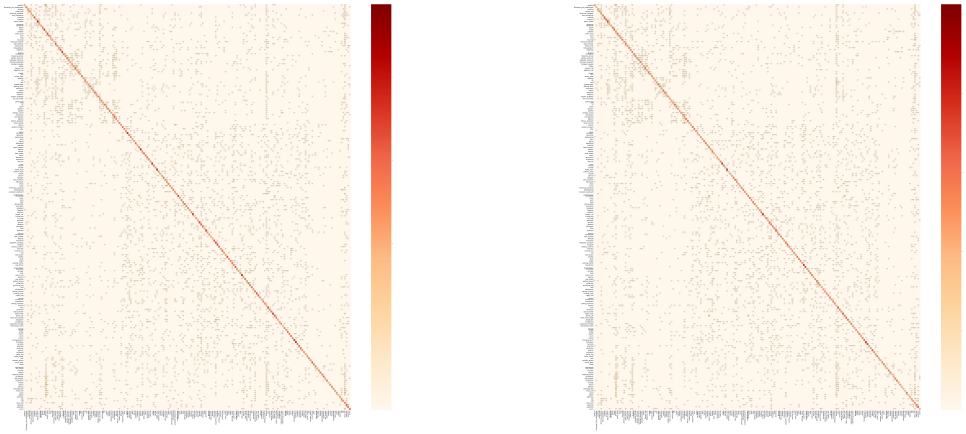

Figure C.7: Comparison of the confusion matrices between the original and pruned versions of CifarNet model on the Tiny-ImageNet dataset.

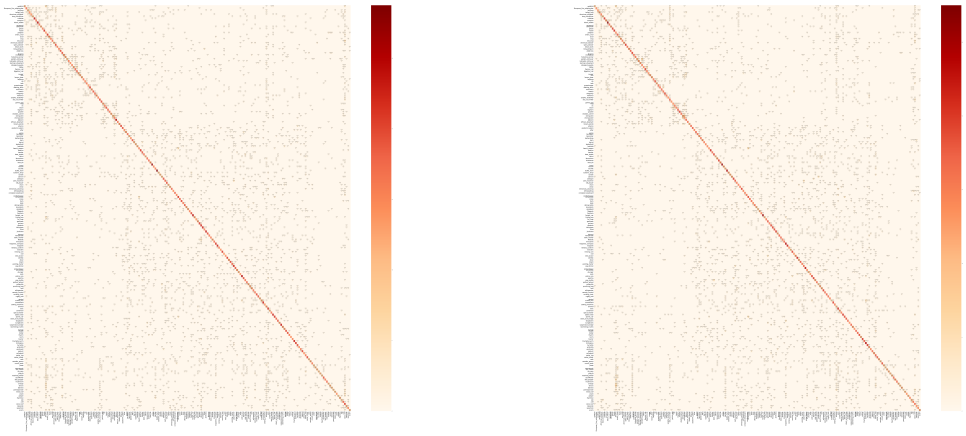

Figure C.8: Comparison of the confusion matrices between the original and pruned versions of AlexNet model on the Tiny-ImageNet dataset.

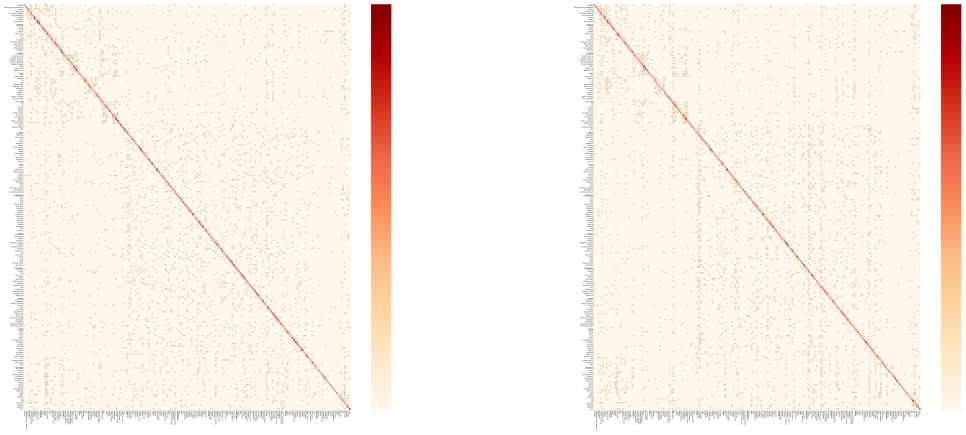

Figure C.9: Comparison of the confusion matrices between the original and pruned versions of VGG16 model on the Tiny-ImageNet dataset.

C. Unsupervised PulseNet Confusion Matrices Results

C. Unsupervised PulseNet Confusion Matrices Results